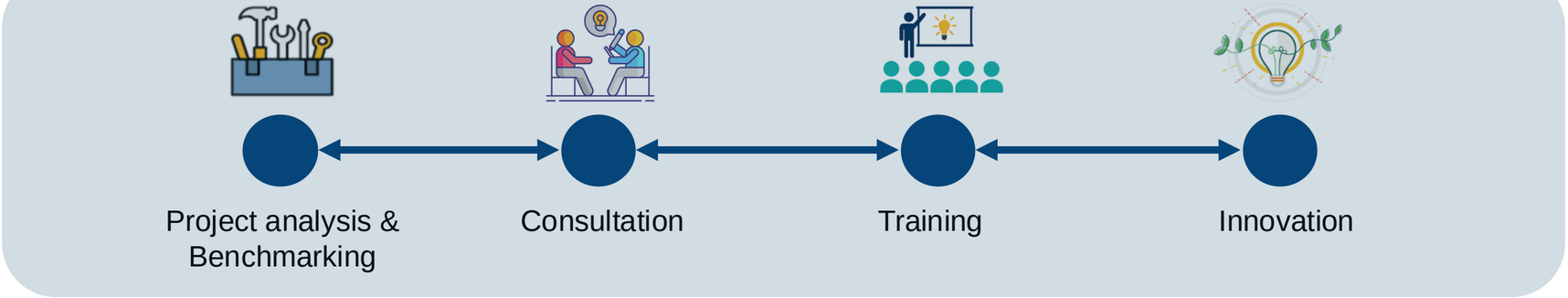
ScRNASeqSnap:
Single cell RNA Seq Snap workflow

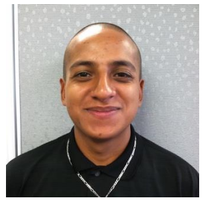
Antonia (Tonia) Chroni, PhD
Senior Bioinformatics Research Scientist
DNB Bioinformatics Core

December 10, 2024

Bioinformatics core, Department of Developmental Neurobiology

Providing advanced bioinformatic services for investigators to leverage omics data





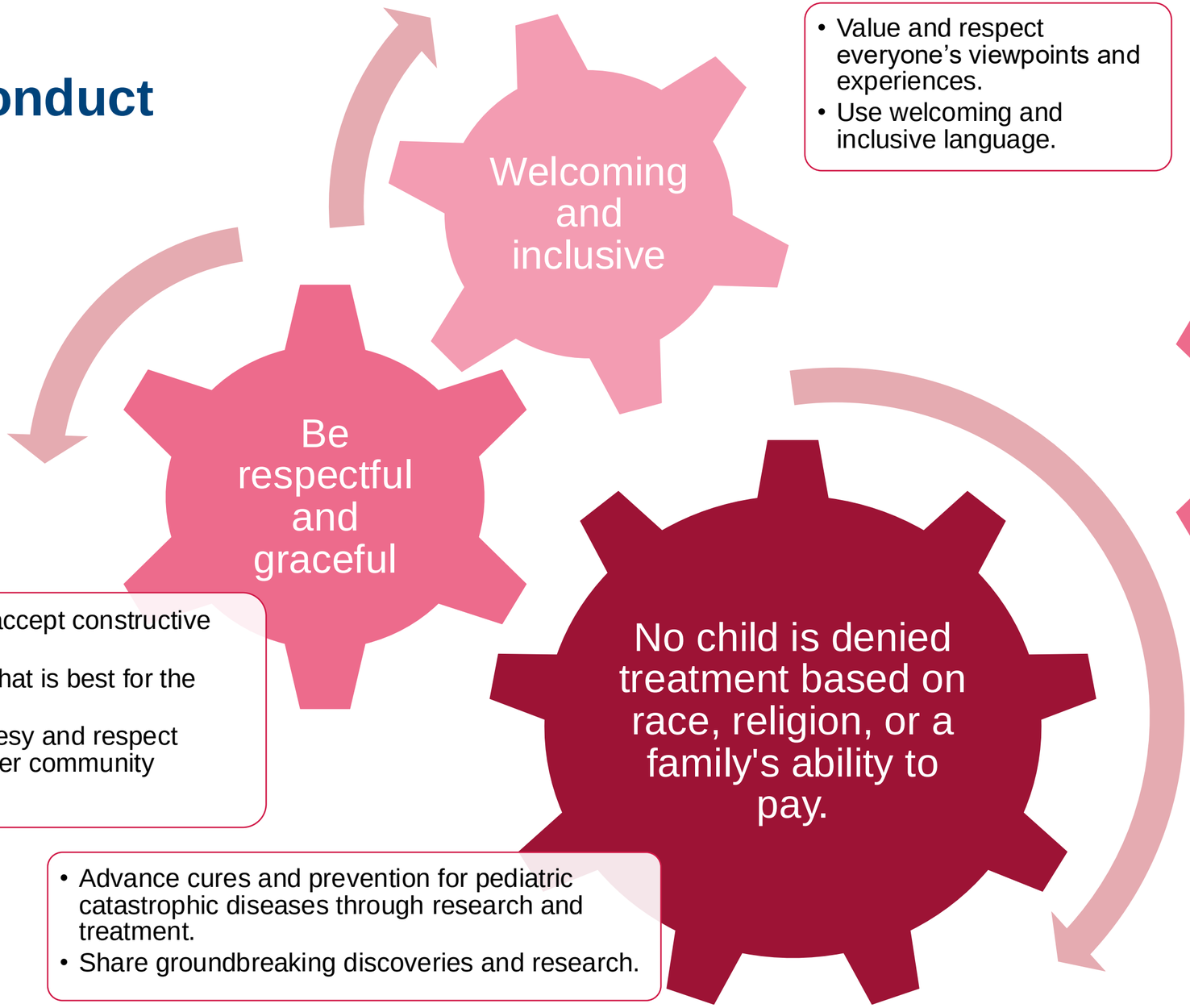
Cody Alexander Ramirez, PhD
Senior Bioinformatics Research Scientist
Core Director
Boston, Massachusetts

Antonia Chroni, PhD
Senior Bioinformatics Research Scientist
New York, New York

Sharon Freshour, PhD
Bioinformatics Research Scientist
St. Louis, Missouri

Asha Jacob Jannu, PhD
Bioinformatics Research Scientist
Indianapolis, Indiana

Code of Conduct



- Value and respect everyone’s viewpoints and experiences.
- Use welcoming and inclusive language.

- We welcome everyone in the research community.
- We are committed to creating a friendly and respectful place for learning, teaching, and contributing.

- Gracefully accept constructive criticism.
- Focus on what is best for the community.
- Show courtesy and respect towards other community members.

- Advance cures and prevention for pediatric catastrophic diseases through research and treatment.
- Share groundbreaking discoveries and research.

Learning Objectives

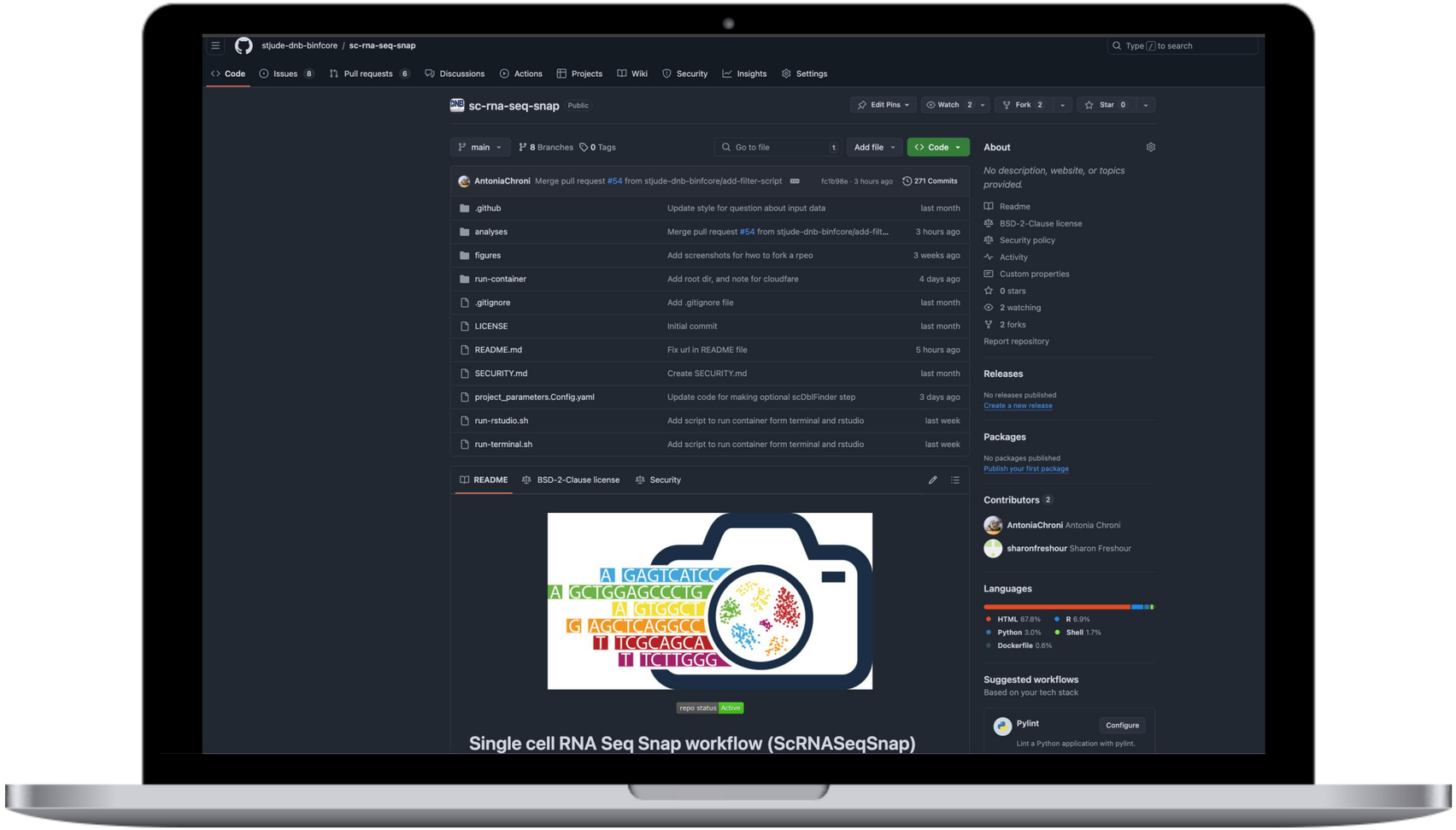
1. Overview of Tools and Methods in the `ScRNASeqSnap` workflow
2. Reviewing Results, Plots, and Reports

Finding cures. Saving children.

Single cell RNA Seq Snap workflow (ScRNASeqSnap)

- 🚀 Automated suite of analysis modules for upstream and downstream analyses
- 🧬 sc/snRNA data analysis from 10X sequencing technology for non-hashed experiments
- 🌮 Human, mouse, and dual index genome experiments
- 📝 Thorough and step-by-step documentation on how to use each analysis module
- 🐳 Reproducible and replicable results via Docker/singularity
- Publicly available at [stjude-dnb-binfcore/sc-rna-seq-snap]

Single cell RNA Seq Snap workflow (ScRNASeqSnap)



https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap

How to access the code

README License Security

To access the code in this repository:

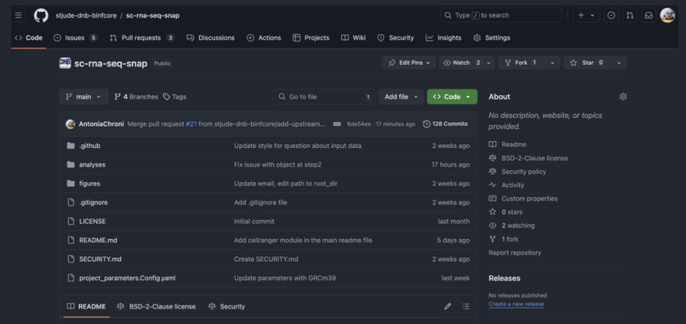
Clone option

1. Clone the repository

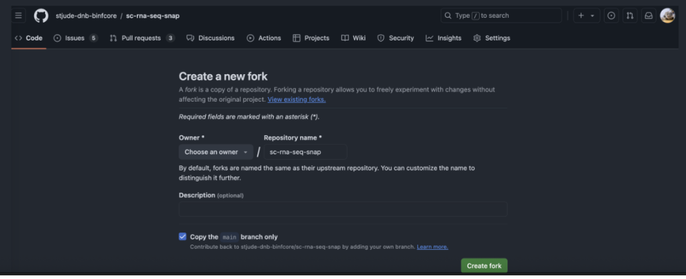
```
git clone https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap.git
```

Fork option

1. Fork the repository on your own account from the main page of the `stjude-dnb-binfcore/sc-rna-seq-snap` by clicking the "Fork" button



2. Change the name if you like, but probably not; click "Create fork"



Finding cures. Saving children.

How to access the code

README License Security

To access the code in this repository:

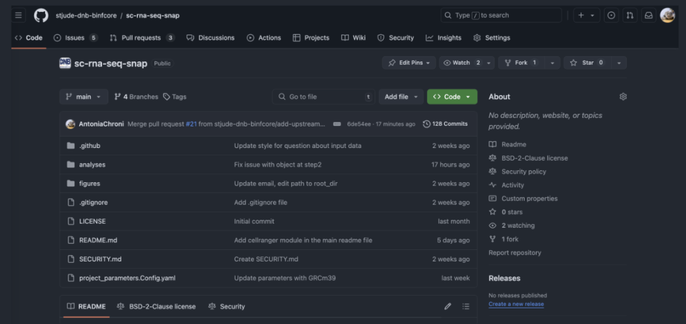
Clone option

1. Clone the repository

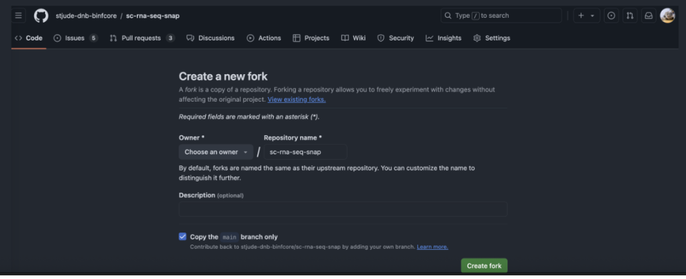
```
git clone https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap.git
```

Fork option

1. Fork the repository on your own account from the main page of the `stjude-dnb-binfcore/sc-rna-seq-snap` by clicking the "Fork" button



2. Change the name if you like, but probably not; click "Create fork"



How to access the code

README License Security

To access the code in this repository:

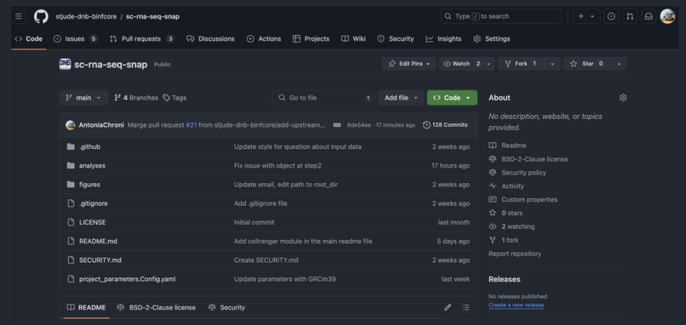
Clone option

1. Clone the repository

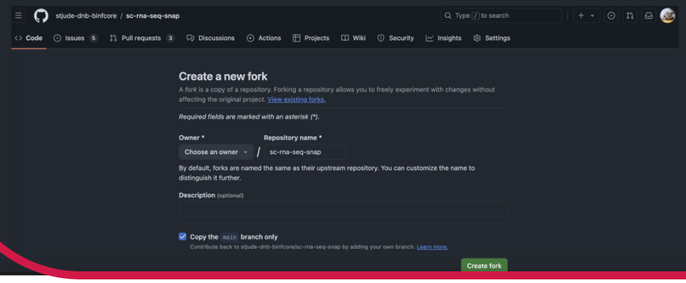
```
git clone https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap.git
```

Fork option

1. Fork the repository on your own account from the main page of the `stjude-dnb-binfcore/sc-rna-seq-snap` by clicking the "Fork" button



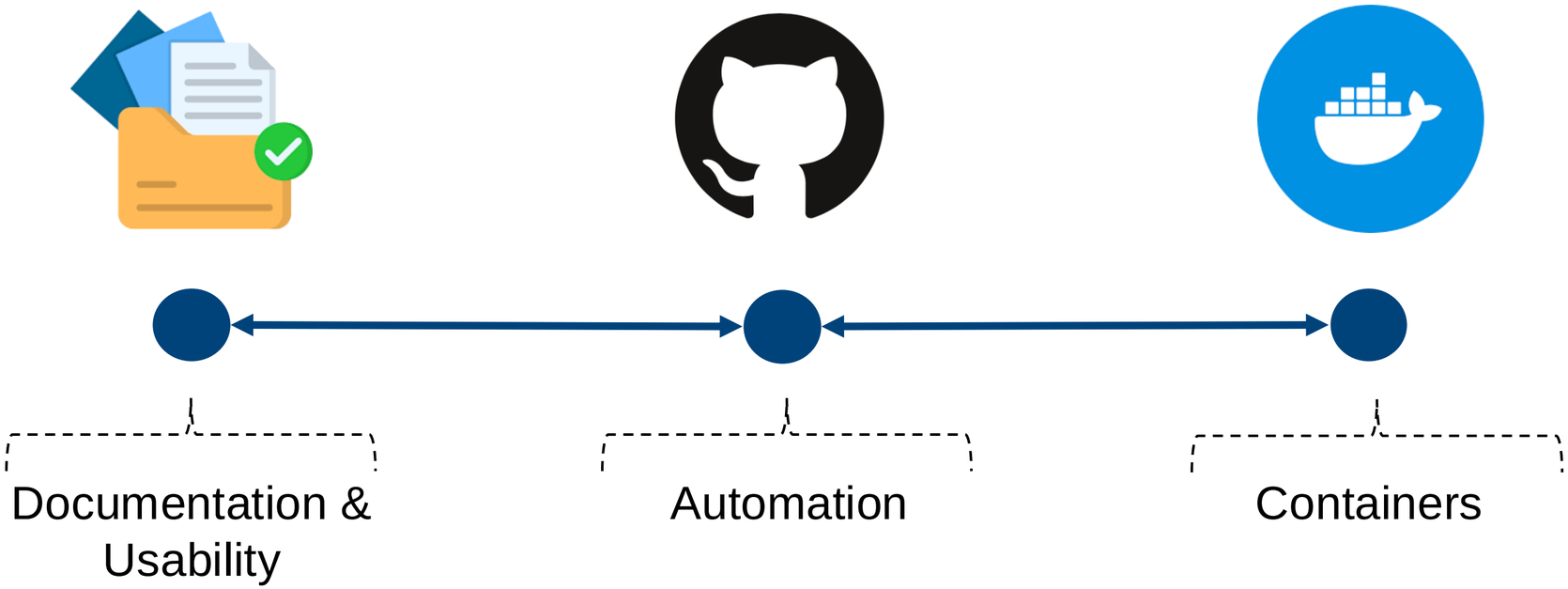
2. Change the name if you like, but probably not; click "Create fork"



Finding cures. Saving children.

ScRNASeqSnap is automated, reproducible and replicable

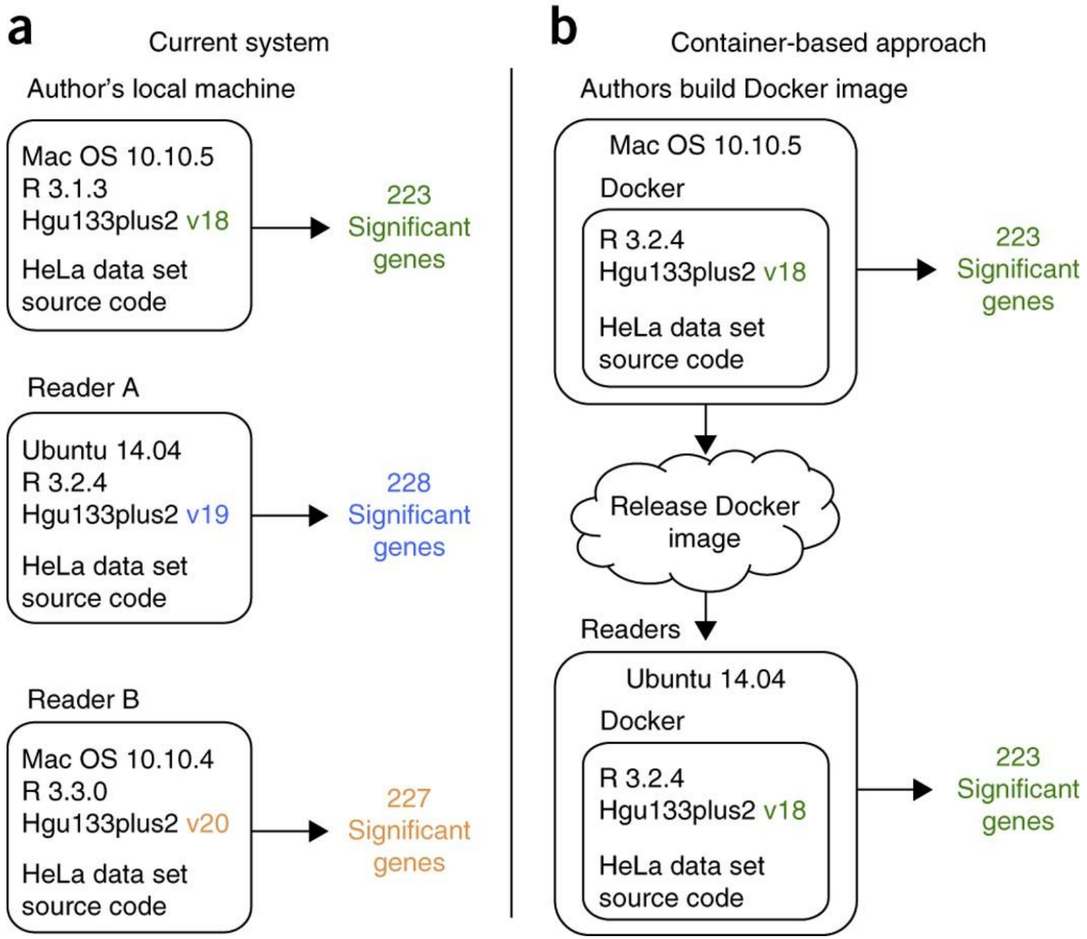
ScRNASeqSnap is automated, reproducible and replicable



Slide from the "Automation and Reproducibility in Computational Biology" course

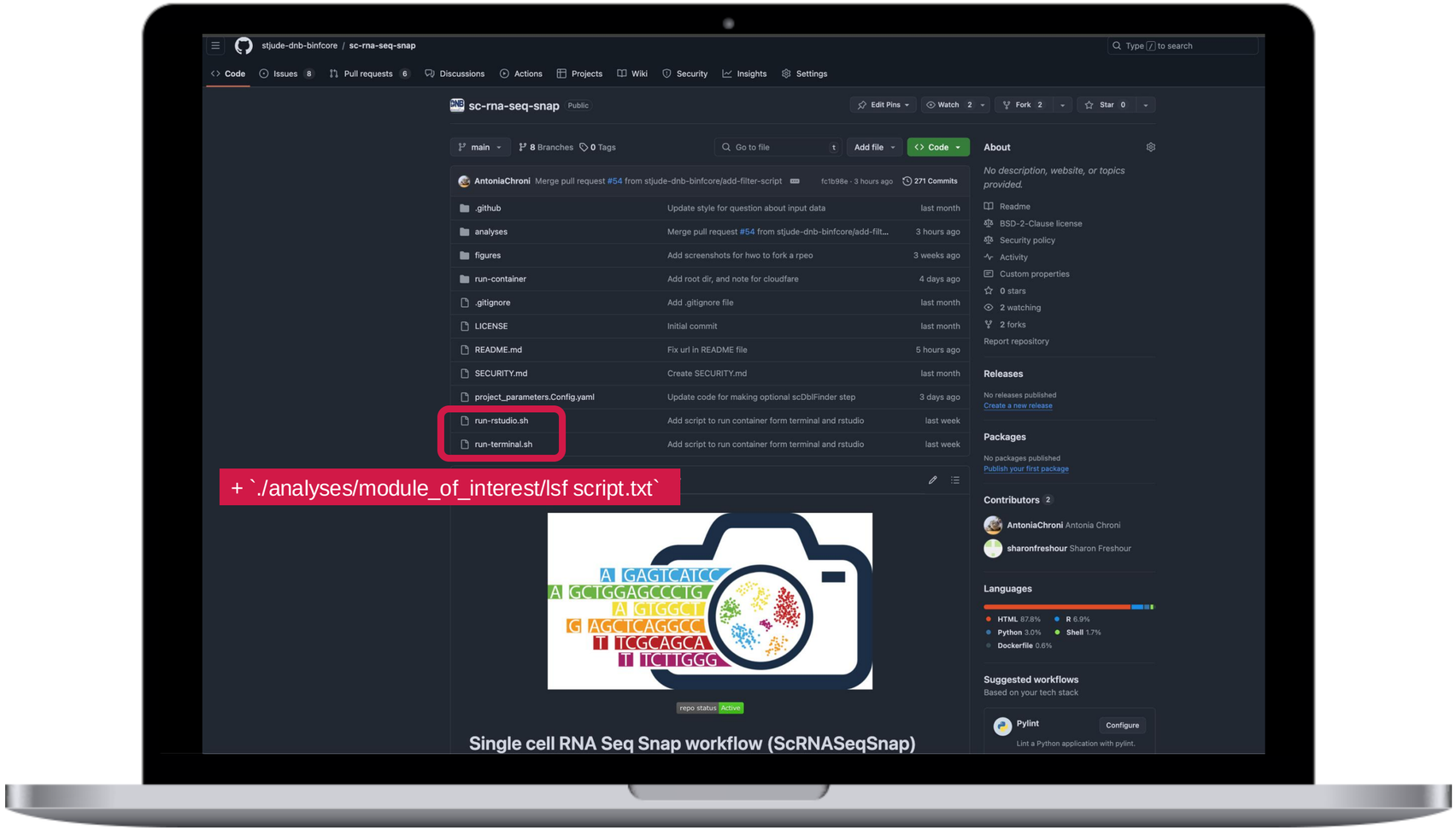
ScRNASeqSnap is automated, reproducible and replicable

- **Dockerfile/rstudio_r_4.4.0_seurat_4.4.0.def**
- Docker and Singularity
  - “Containers” that include everything from the operating system up
  - Run one OS inside another, with all the things frozen to particular versions.



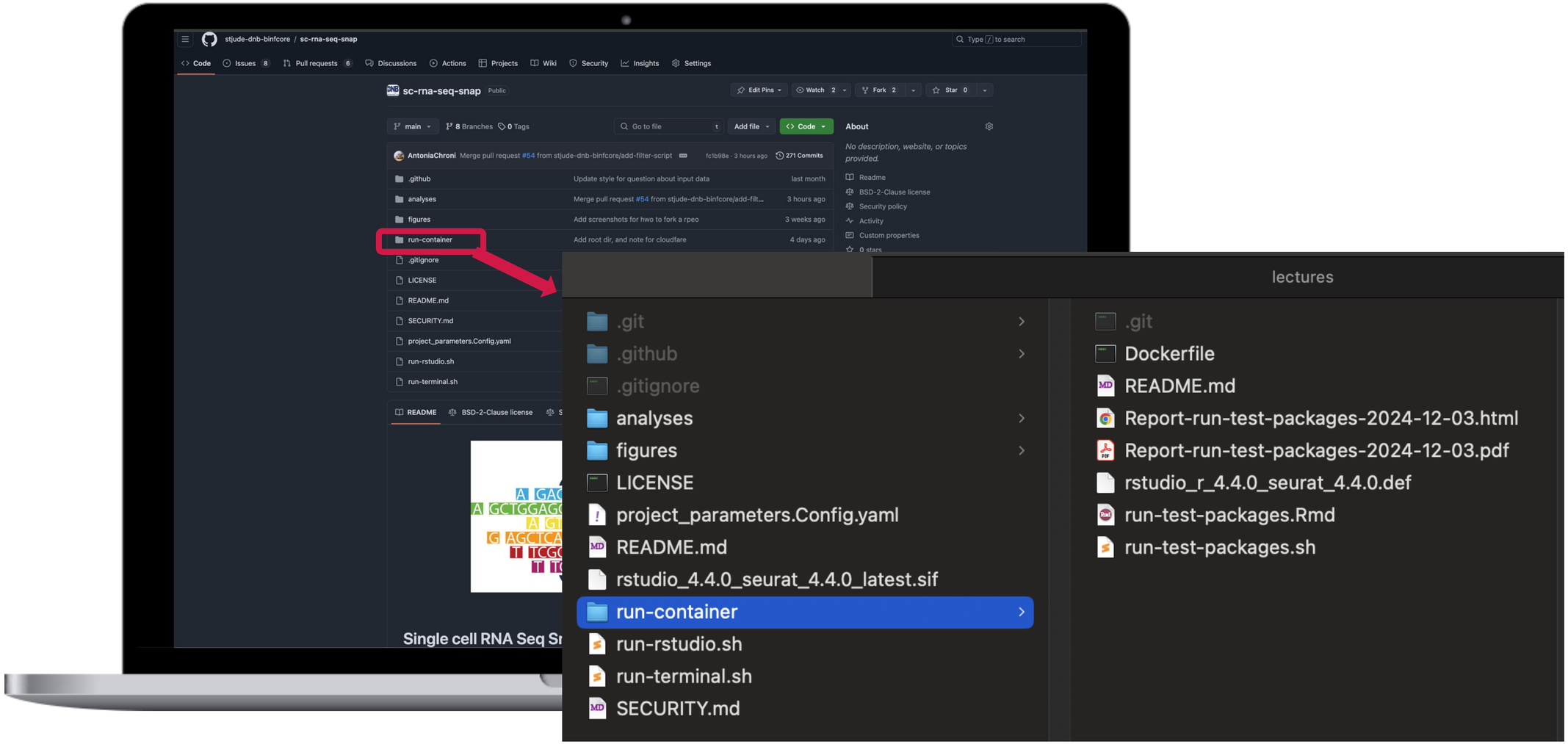
Slide from the ”Automation and Reproducibility in Computational Biology” course

ScRNASeqSnap is automated, reproducible and replicable



https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap

ScRNASeqSnap is automated, reproducible and replicable

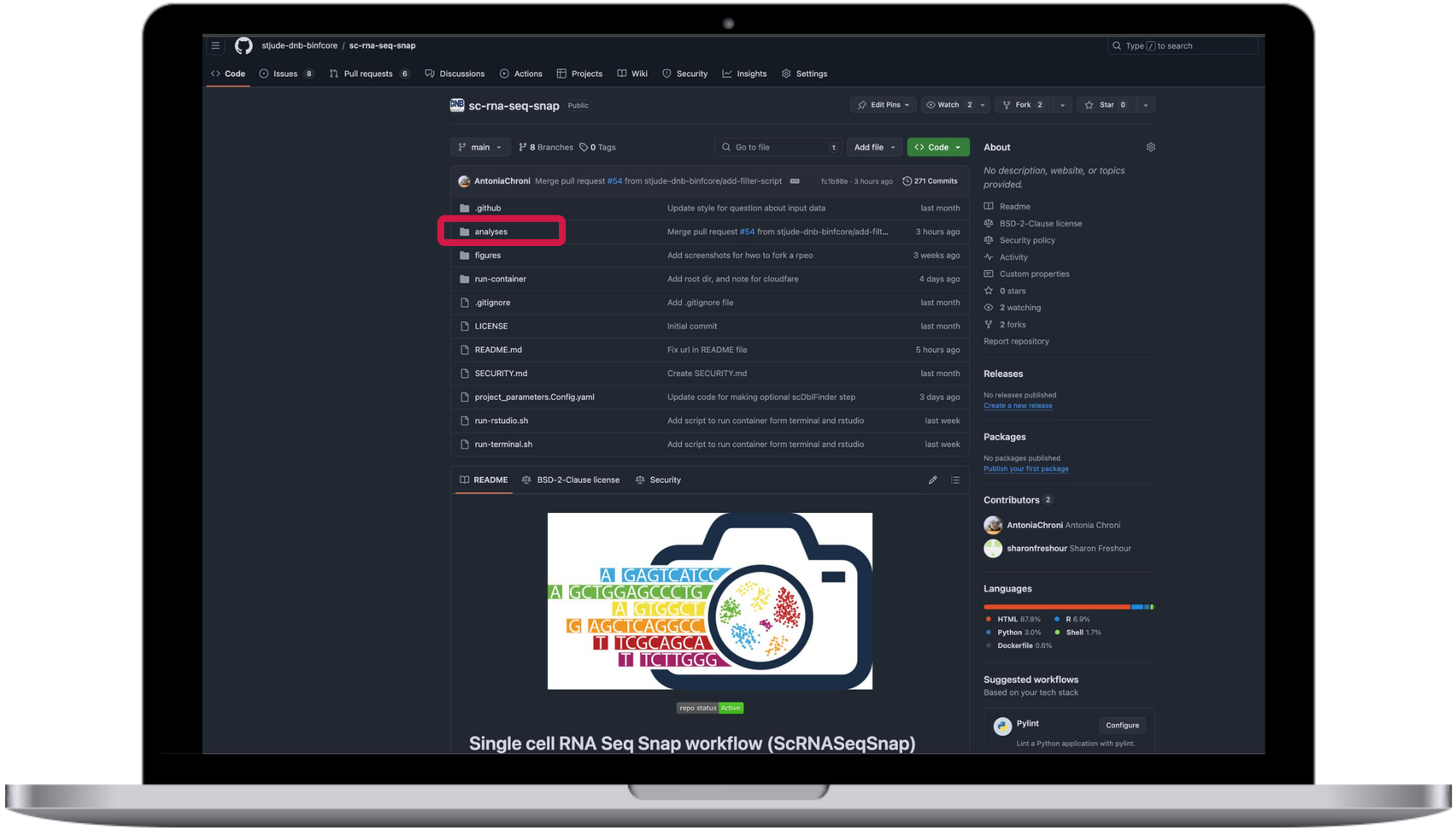


Repository organization and project tidiness

- **Use Folders/Directories**
- **Keep separate projects separated**
- **Separate sections for units within a project**
  - analyses/<module_name>
    - code
    - plots
    - results
  - data
  - figures
  - LICENSE
  - project_parameters.Config.yaml
  - SECURITY.md
  - …
- **Documentation throughout:** Describe what files do and how they are organized/folder.

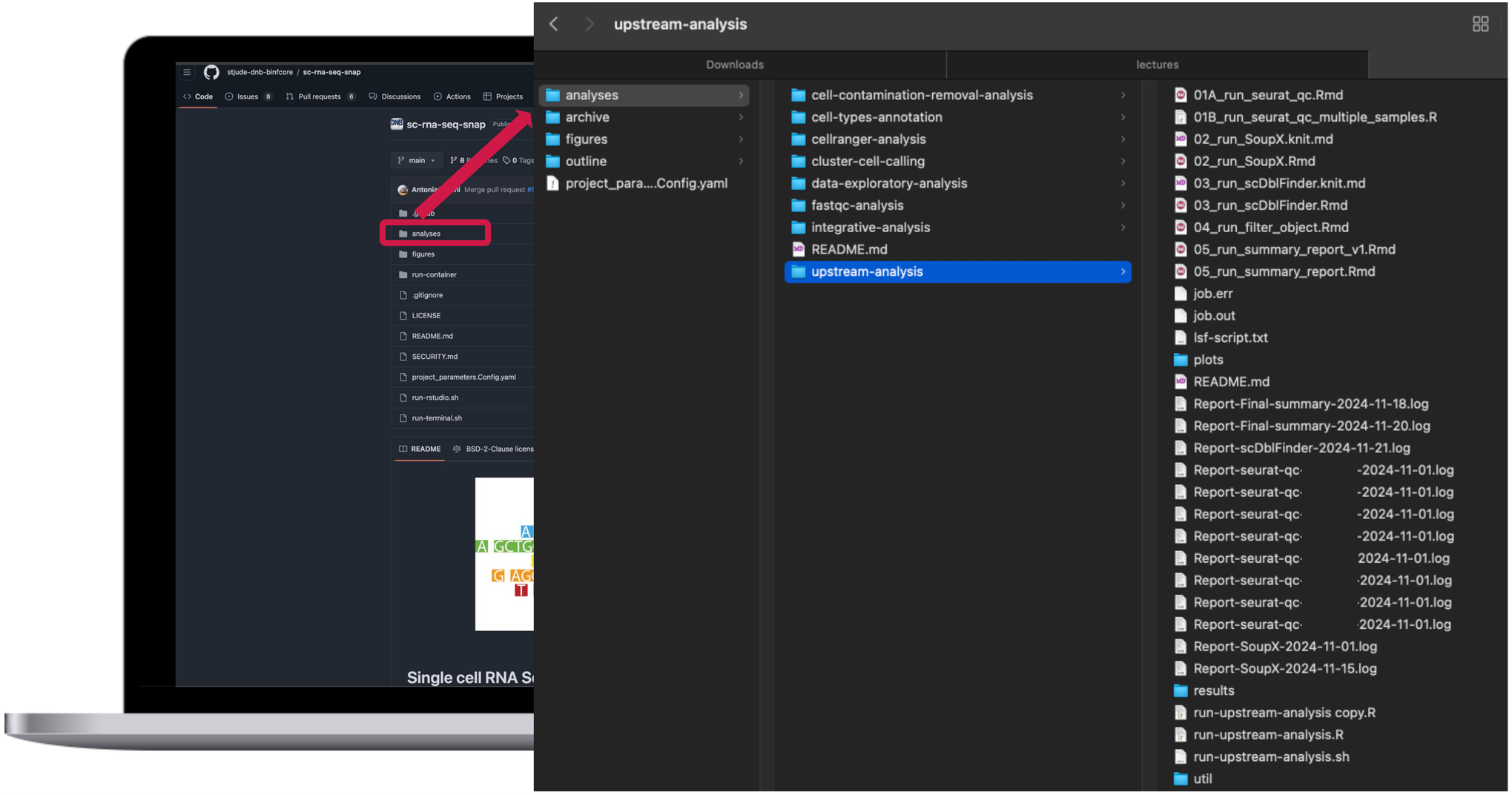
Slide from the "Automation and Reproducibility in Computational Biology" course

Repository organization and project tidiness

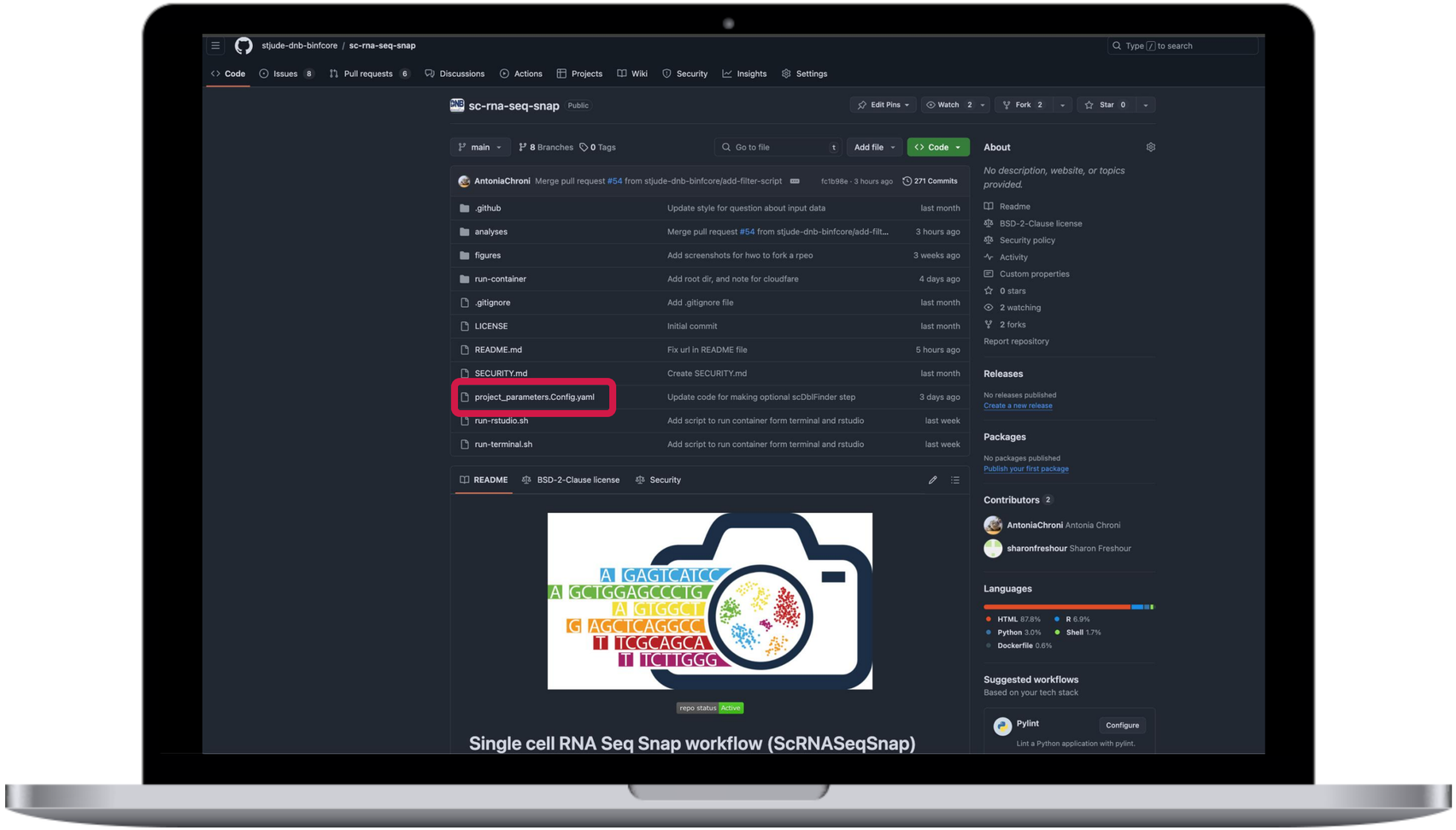


https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap

My typical project organization system



Customize your project based on the experiment!



https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap

Project parameters file

project_parameters.Config.yaml

```yaml
# the following parameters are the same across the project and might be needed in more than one module #
root_dir: "./sc-rna-seq-snap" # path to the main dir of the project where GitHub repo lives
data_dir: "./sc-rna-seq-snap/analyses/cellranger-analysis/results/02_cellranger_count/ForcedCells8000Parameters" # path to data dir of the proj
metadata_dir: "./mouse-test-dataset" # path to metadata dir of the project
genome_name: "GRCm39" # define genome reference and versioning
PROJECT_NAME: "mouse-test-dataset"
PI_NAME: "Stanislav Zakharenko"
TASK_ID: "NA"
PROJECT_LEAD_NAME: "NA"
DEPARTMENT: "Developmental Neurobiology"
LEAD_ANALYSTS: "Antonia Chroni, PhD"
GROUP_LEAD: "Cody A. Ramirez, PhD"
CONTACT_EMAIL: "antonia.chroni@stjude.org"
PIPELINE: "Standard sc-/sn-RNA-Seq Analysis in 10X Genomics data"
START_DATE: "10/15/2024"
COMPLETION_DATE: "ONGOING"


# the following parameters are set up as default values and/or are specific for the following modules:
# `./analyses/fastqc-analysis`
fastqc_dir: "." #path to the fastqc files for the `fastqc-analysis` module

# `./analyses/cellranger-analysis`
genome_reference_path: "./" #path to genome reference to be used for the `cellranger-analysis` module
cellranger_parameters: "ForcedCells8000Parameters" # or "ForcedCells8000Parameters" etc
genome_name_cellranger: "GRCm39" #please define the genome of preference for dual genomes. In case for single genomes, please use the same as u

# `./analyses/upstream-analysis`
print_pdf_seurat_multiple_samples: "YES" # set value for 01B_run_seurat_qc_multiple_samples.R
use_condition_split_seurat_multiple_samples: "NO" # set value for 01B_run_seurat_qc_multiple_samples.R
grouping: "orig.ident" # define grouping to use
Regress_Cell_Cycle_value: "NO" # Indicates whether or not to regress for cell cycle and, if so, which method to use and scale data; acceptable
assay: "RNA" # define assay
min_genes: 300 # define minimum number of genes for filtering
min_count: 500 # define minimum number of UMIs for filtering
mtDNA_pct_default: 10 # define minimum percentage of mtDNA for filtering
normalize_method: "log_norm" # define method for normalization of counts
num_pcs: 30 # define number of principal components
nfeatures_value: 3000 # define number of variable features
prefix: "lognorm" # create label based on the normalization method used
use_miQC: "NO" # define use of miQC R package or not; see `README.md` file for more information; acceptable values: "YES" and "NO"
use_only_step1: "YES" # define use of both or only first step for filtering low quality cells; see `README.md` file for more information; accep
condition_value: "Genotype" # define main condition of the project; this can be used for visualization purposes on the UMAPs; value to be extra
num_dim_seurat_qc: [20, 25] # number of PCs to use in UMAP
num_neighbors_seurat_qc: [10, 20, 30] # number of neighbors to use in UMAP
soup_fraction_value_default: 0.05 # set rho default value to use if estimated rho is > 20%
num_dim_filter_object: 30 # set one value for 04_run_filter_object.Rmd
num_neighbors_filter_object: 30 # set one value for 04_run_filter_object.Rmd
use_SoupX_filtering_filter_object: "YES" # set for 04_run_filter_object.Rmd
use_condition_split_filter_object: "YES" # set for 04_run_filter_object.Rmd
print_pdf_filter_object: "NO" # set for 04_run_filter_object.Rmd
```

How to run the code

README License Security

project_parameters.Config.yaml Update parameters with GRCm39 last week

To run the code in this repository:

1. Replace the `project_parameters.Config.yaml` with your file paths and parameters.
2. Navigate to an analysis module and run the shell script of interest:

```
cd ./sc-rna-seq-snap/analyses/<module_of_interest>
```

If you have forked the repo, you will need to do the following steps before running the script of interest. **If you have cloned the repo, you can skip this.**

You need to do `sync fork` of your project repo at GitHub before running a module, if your branch is behind the main branch of the `stjude-dnb-binfcore/sc-rna-seq-snap:main`. This will update the main branch of your project repo with the new code and modules (if any). This will add code and not break any analyses already run in your project repo.

Then navigate to your `./sc-rna-seq-snap` project repo and ensure you are at the `main` branch. If not, you will need to `git checkout` to the main branch.

```
git branch
git checkout main
```

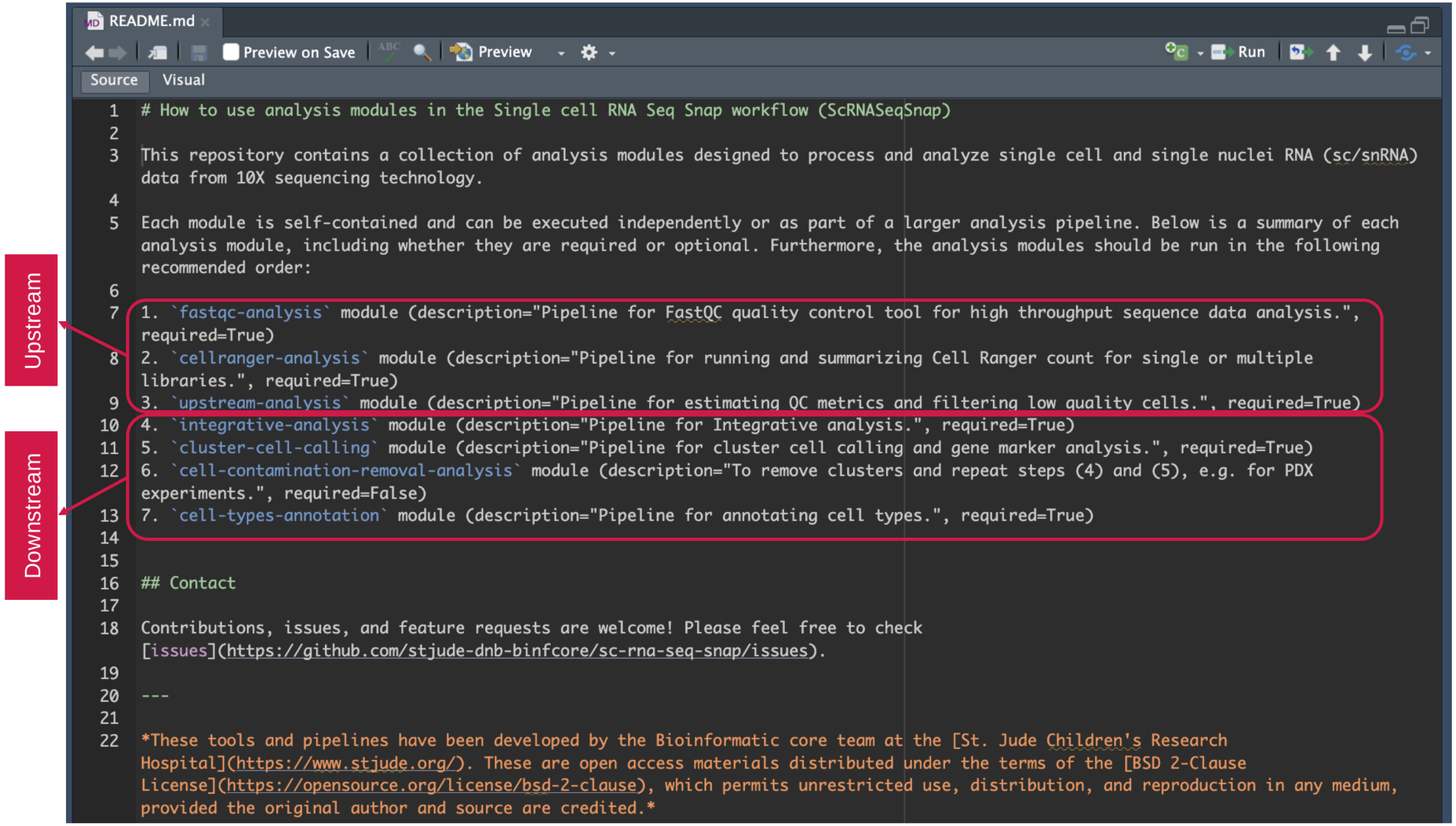
Finally, `git pull` to get the most updated changes and code in your project repo.

```
git pull
```

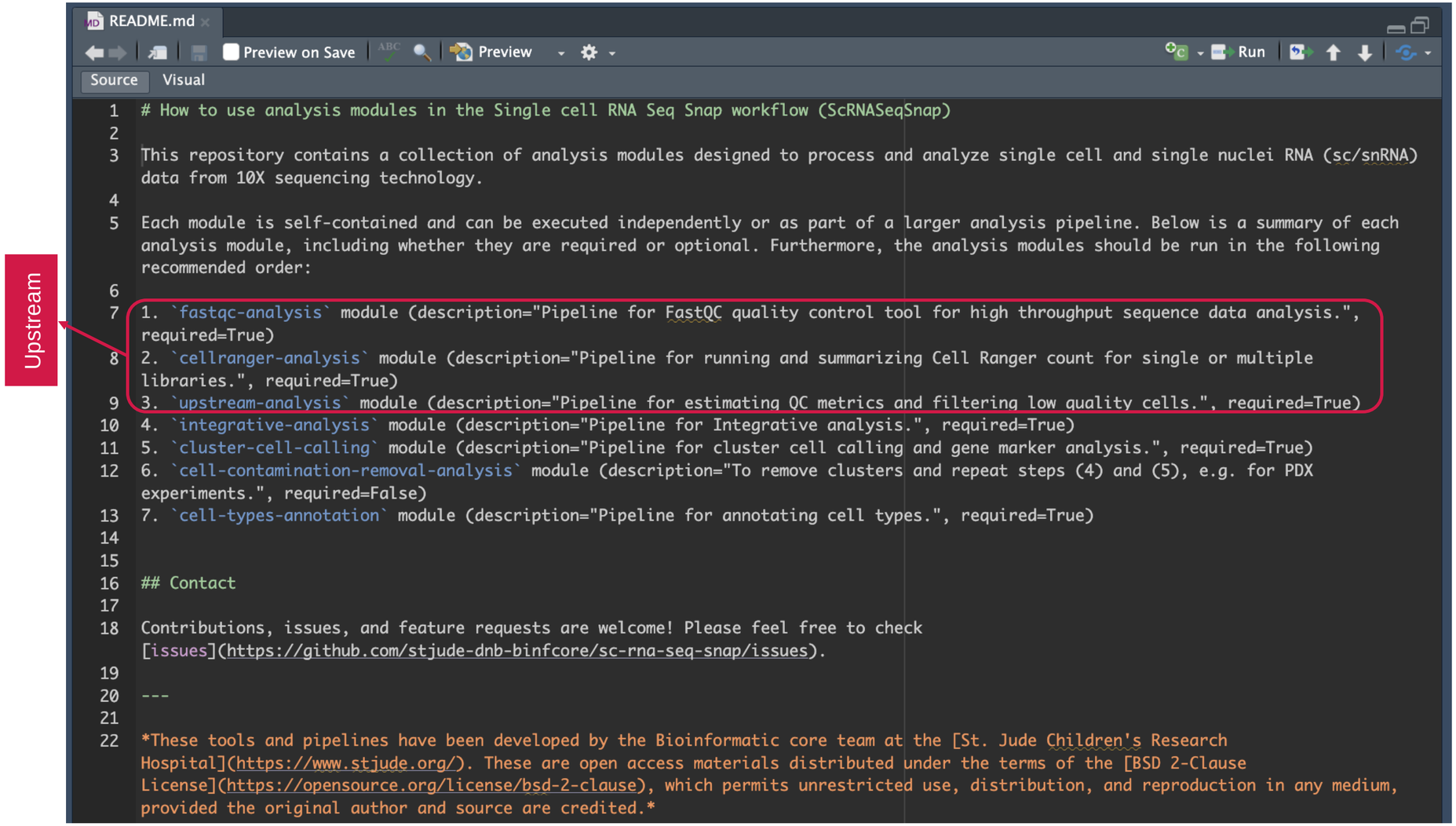
Below is the main directory structure listing the analyses and data files used in this repository

```
├── analyses
|   ├── cellranger-analysis
|   ├── fastqc-analysis
|   └── upstream-analysis
├── figures
├── LICENSE
├── project_parameters.Config.yaml
├── README.md
└── SECURITY.md
```

ScRNASeqSnap analysis modules

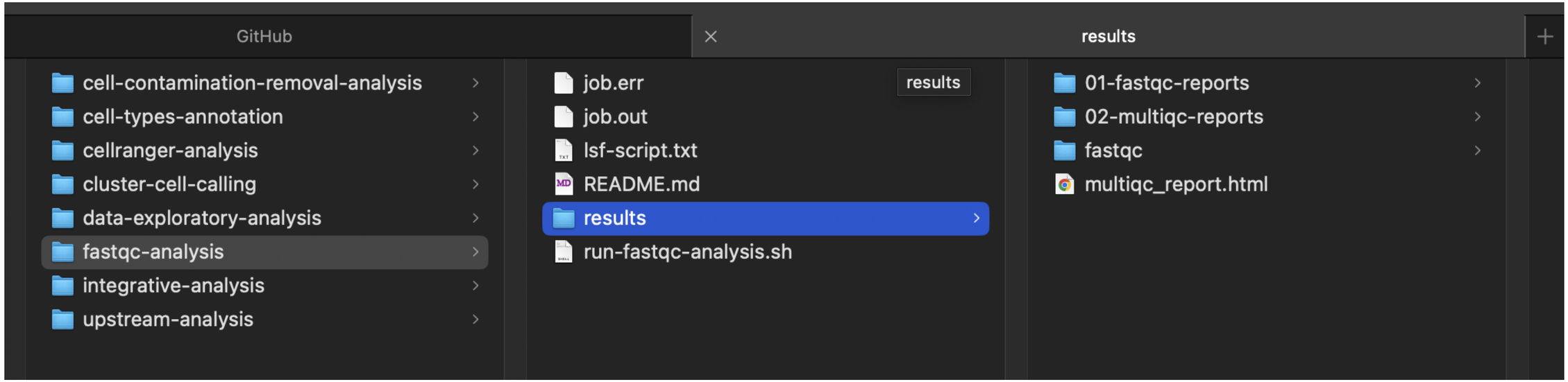


ScRNASeqSnap analysis modules



`fastqc-analysis` module

description="Pipeline for FastQC quality control tool for high throughput sequence data analysis.", required=True



`run-fastqc-analysis.sh`

run-fastqc-analysis.sh

Run | Run Script

```
#!/bin/bash

set -e
set -o pipefail

# set up running directory
cd "$(dirname "${BASH_SOURCE[0]}")"

######################################################################
# Read multiple values and assign them to variables by parsing yaml file
fastqc_dir=$(cat ../../project_parameters.Config.yaml | grep 'fastqc_dir:' | awk '{print $2}')
fastqc_dir=${fastqc_dir//\"/}  # Removes all double quotes

# Create results dir
mkdir -p results
mkdir -p results/01-fastqc-reports

#############################################################################################################
# Run FastQC per library
fastqc -o results/01-fastqc-reports ${fastqc_dir}/*/*R2*.fastq.gz

#############################################################################################################
# Run multiqc for all samples in the `results` dir
# to summarize results

cd results/01-fastqc-reports
multiqc .

# rename folder
mv multiqc_data 02-multiqc-reports

# move the files related to multiqc in the main `results` dir
mv 02-multiqc-reports ../
mv multiqc_report.html ../

```


Finding cures. Saving children.

`run-fastqc-analysis.sh` and `lsf-script.txt`

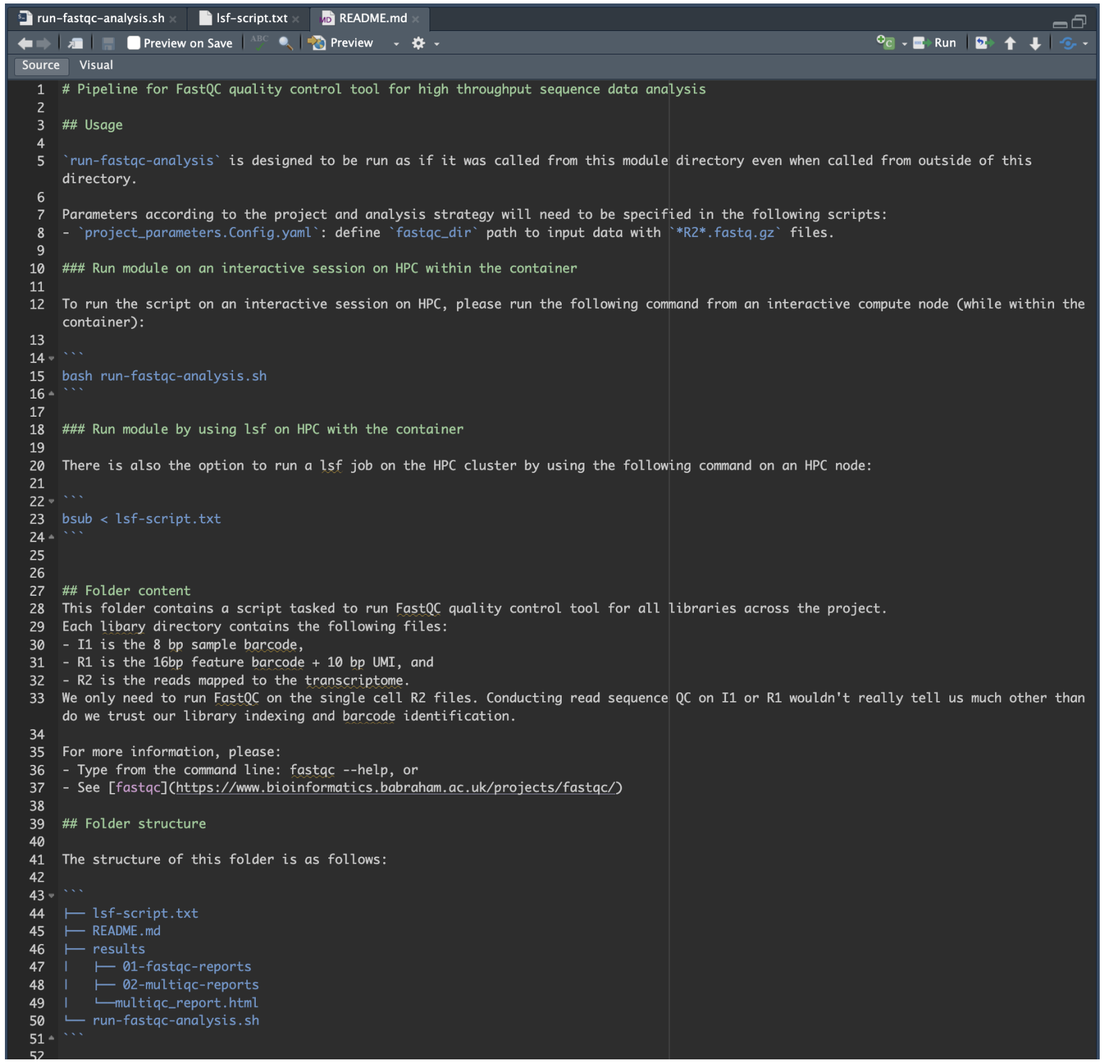
run-fastqc-analysis.sh

```bash
#!/bin/bash

set -e
set -o pipefail

# set up running directory
cd "$(dirname "${BASH_SOURCE[0]}")"

###################################################################
# Read multiple values and assign them to variables by parsing yaml file
fastqc_dir=$(cat ../../project_parameters.Config.yaml | grep 'fastqc_dir:' | awk '{print $2}')
fastqc_dir=${fastqc_dir//\"/}  # Removes all double quotes

# Create results dir
mkdir -p results
mkdir -p results/01-fastqc-reports

#####################################################################################################
# Run FastQC per library
fastqc -o results/01-fastqc-reports ${fastqc_dir}/*/*R2*.fastq.gz

#####################################################################################################
# Run multiqc for all samples in the `results` dir
# to summarize results

cd results/01-fastqc-reports
multiqc .

# rename folder
mv multiqc_data 02-multiqc-reports

# move the files related to multiqc in the main `results` dir
mv 02-multiqc-reports ../
mv multiqc_report.html ../

```

lsf-script.txt

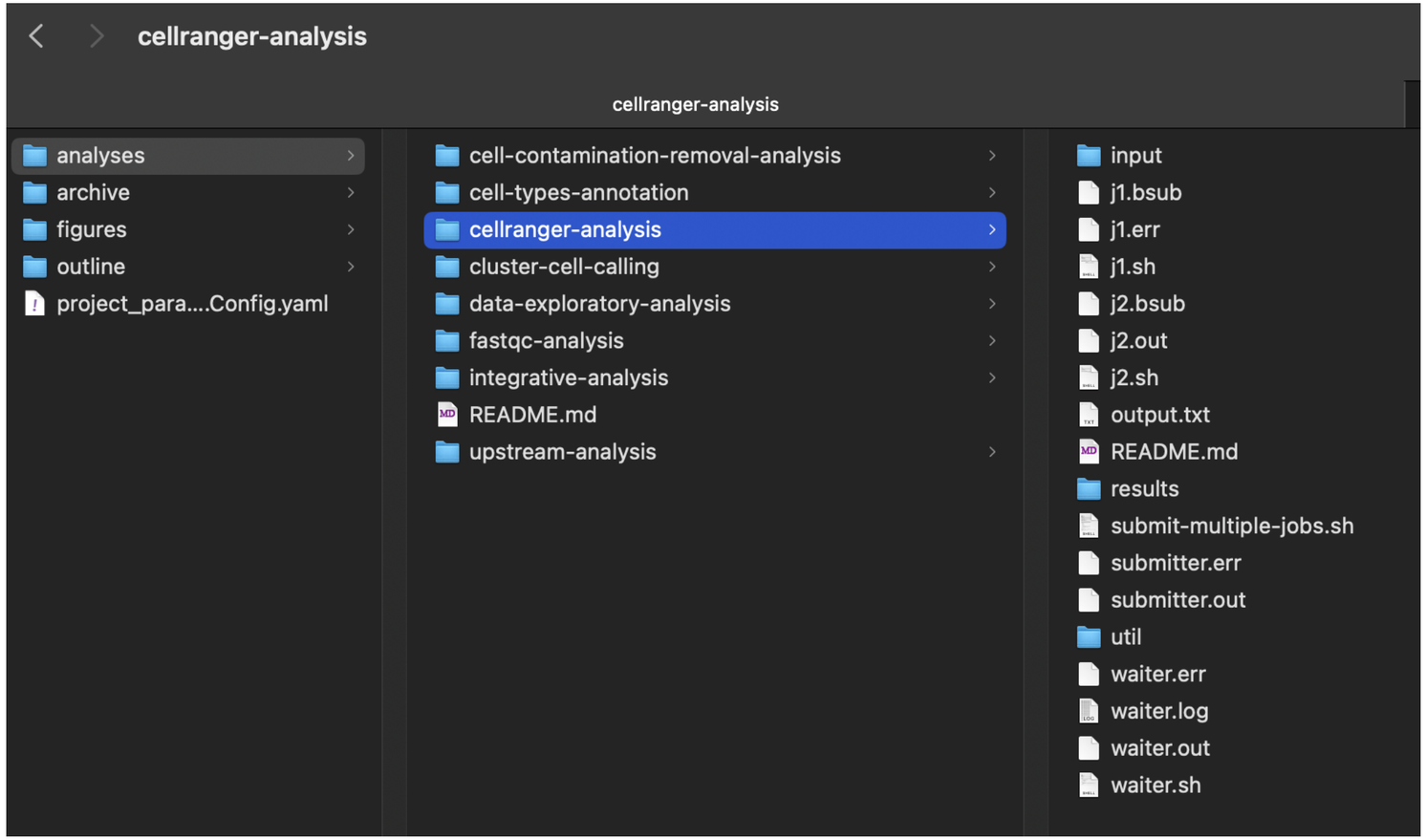
```
#BSUB -P project
#BSUB -J fastqc-analysis
#BSUB -oo job.out -eo job.err
#BSUB -n 2
#BSUB -R "rusage[mem=4GB] span[hosts=1]"
#BSUB -cwd "."

singularity exec ../../rstudio_4.4.0_seurat_4.4.0_latest.sif bash run-fastqc-analysis.sh

```

README.md


run-fastqc-analysis.sh
lsf-script.txt
README.md
Preview on Save
Preview
Run
Source
Visual
Pipeline for FastQC quality control tool for high throughput sequence data analysis
Usage
`run-fastqc-analysis` is designed to be run as if it was called from this module directory even when called from outside of this directory.
Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `project_parameters.Config.yaml`: define `fastqc_dir` path to input data with `*R2*.fastq.gz` files.
Run module on an interactive session on HPC within the container
To run the script on an interactive session on HPC, please run the following command from an interactive compute node (while within the container):
```
bash run-fastqc-analysis.sh
```
Run module by using lsf on HPC with the container
There is also the option to run a lsf job on the HPC cluster by using the following command on an HPC node:
```
bsub < lsf-script.txt
```
Folder content
This folder contains a script tasked to run FastQC quality control tool for all libraries across the project.
Each libary directory contains the following files:
- I1 is the 8 bp sample barcode,
- R1 is the 16bp feature barcode + 10 bp UMI, and
- R2 is the reads mapped to the transcriptome.
We only need to run FastQC on the single cell R2 files. Conducting read sequence QC on I1 or R1 wouldn't really tell us much other than do we trust our library indexing and barcode identification.
For more information, please:
- Type from the command line: fastqc --help, or
- See [fastqc](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
Folder structure
The structure of this folder is as follows:
```
├── lsf-script.txt
├── README.md
├── results
|   ├── 01-fastqc-reports
|   ├── 02-multiqc-reports
|   └──multiqc_report.html
└── run-fastqc-analysis.sh
```

`cellranger-analysis` module

description="Pipeline for running and summarizing Cell Ranger count for single or multiple libraries.", required=True



`cellranger-analysis` module

README.md

Preview on Save | Preview | Run

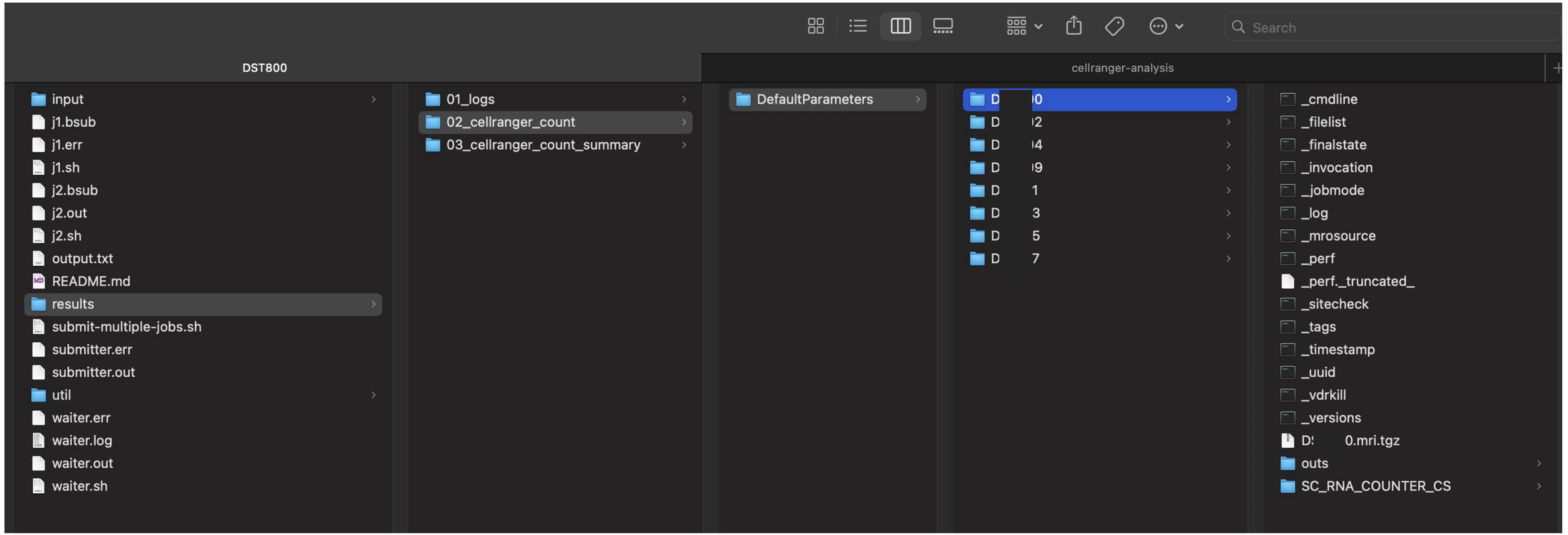
Source | Visual

```
# Pipeline for running and summarizing Cell Ranger count for single or multiple libraries for sc-/sn-RNA-Seq Analysis in 10X Genomics data

## Usage

To run all the scripts in this module from the command line sequentially, use:

```
bsub < submit-multiple-jobs.sh
```

The `submit-multiple-jobs.sh` script is designed to be run as if it was called from this module directory even when called from outside of this directory.
   - Step 1: To run the `j1.sh` script to align single or multiple libraries in parallel, i.e., `run-cellranger-analysis`.
   - Step 2: To run `j2.sh` to summarize alignment results, i.e., `summarize-cellranger-analysis`. The latter script will be on hold and executed once all libraries are aligned and `j1.sh` is complete. This is been taken care of by `waiter.sh` script.

Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `../../project_parameters.Config.yaml`: define the `metadata_dir`, `genome_reference_path`, `cellranger_parameters`, and `genome_name_cellranger`. Please submit an [issue](https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap/issues) to request the path to the reference genome of preference. Our team at the Bioinformatics core at DNB maintains the following genome references: `GRCh38`, `GRCm39`, `GRCh38ANDGRCm39`, and `GRCh38_GFP_tdTomato`, for human, mouse, and dual index genomes, respectively. Otherwise, specify the path to the reference genome of your preference.

- `submit-multiple-jobs.sh`: Here, the user will need to set up the absolute path for the directory in `#BSUB -cwd` and `prefix` parameters.

- `waiter.sh`: Here, the user will need to set up the absolute path for the directory in `#BSUB -cwd` and `prefix` parameters. Also, user needs to replace `DST` with the Sample ID used for the samples of the project. This requires that all of the Sample IDs are the same, e.g., DST800, DST802, DST811 and so on.

- `j1.sh`:
   - `--file`: This is defined by default in the `./sc-rna-seq-snap/analyses/cellranger-analysis/input/` dir. If the file lives in another folder, e.g., `metadata_dir`, this should be modified accordingly in the `../../project_parameters.Config.yaml`. The file needs to be in `*.txt` file format. In addition, there is a code line to convert `*.tsv` file to `*.txt` file, if needed. If the file exists already in `./input` as `*.txt`, this line of code will be ignored. This file can contain either one or multiple samples as long as the following minimum parameters are provided: `ID`, `SAMPLE`, and `FASTQ` columns.
  - `--force_cells`: User can add flags as necessary for their analyses and compare alignments with CellRanger, e.g., by using `--force_cells=8000` to constraint the number of cells to be used for alignment. We recommend to run by default, and after careful assessment to edit parameters. We have found that the default parameters set up here work well for most of the cases.

- `j2.sh`:
   - In case of dual genomes in the experiment, user will need to add the `--genome ${genome_name_cellranger}` flag to summarize results generated by CellRanger for all libraries.

Default memory for running the alignment is set up to 16GB. In case more memory is needed to run a specific sample, this can be modified here:
- `./util/run_cellranger.py`: User can search the following and change accordingly `-n 4 -R "rusage[mem=4000] span[hosts=1]`, if that is necessary. However, we have found that these resources work well for most data.
```

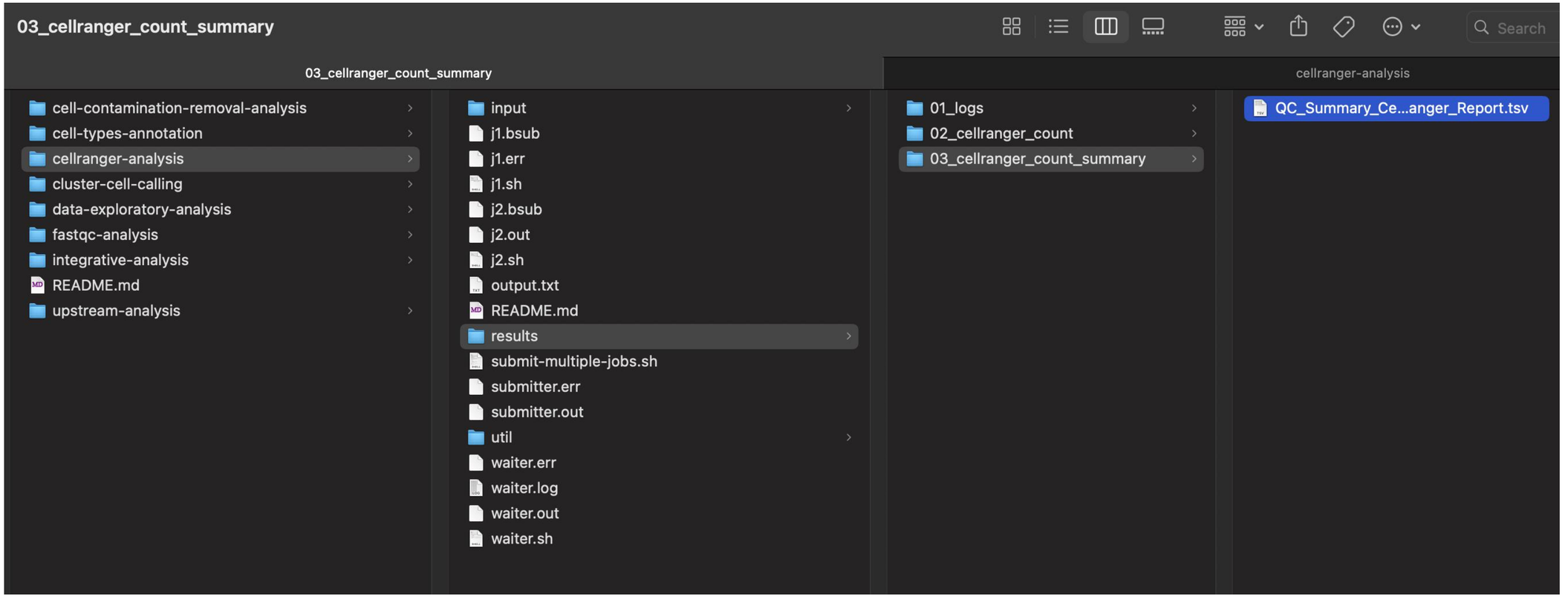
`cellranger-analysis` module

```
## Folder content
This folder contains scripts tasked to run and summarize Cell Ranger count for single or multiple libraries for sc-/sn-RNA-Seq Analysis in 10X Genomics data across the project. For more information and updates, please see [Cell Ranger support page](https://www.10xgenomics.com/support/software/cell-ranger/latest/analysis/running-pipelines/cr-gex-count).

This module uses CellRanger v8.0.1 for the alignment.


## Folder structure

The structure of this folder is as follows:

```
├── input
| └── project_metadata.txt
├── j1.sh
├── j2.sh
├── README.md
├── results
| ├── 01_logs
| ├── 02_cellranger_count
| | └── DefaultParameters
| └── 03_cellranger_count_summary
├── submit-multiple-jobs.sh
├── util
| ├── run_cellranger.py
| └── summarize_cellranger_results.py
└── waiter.sh
```
```

`cellranger-analysis` module

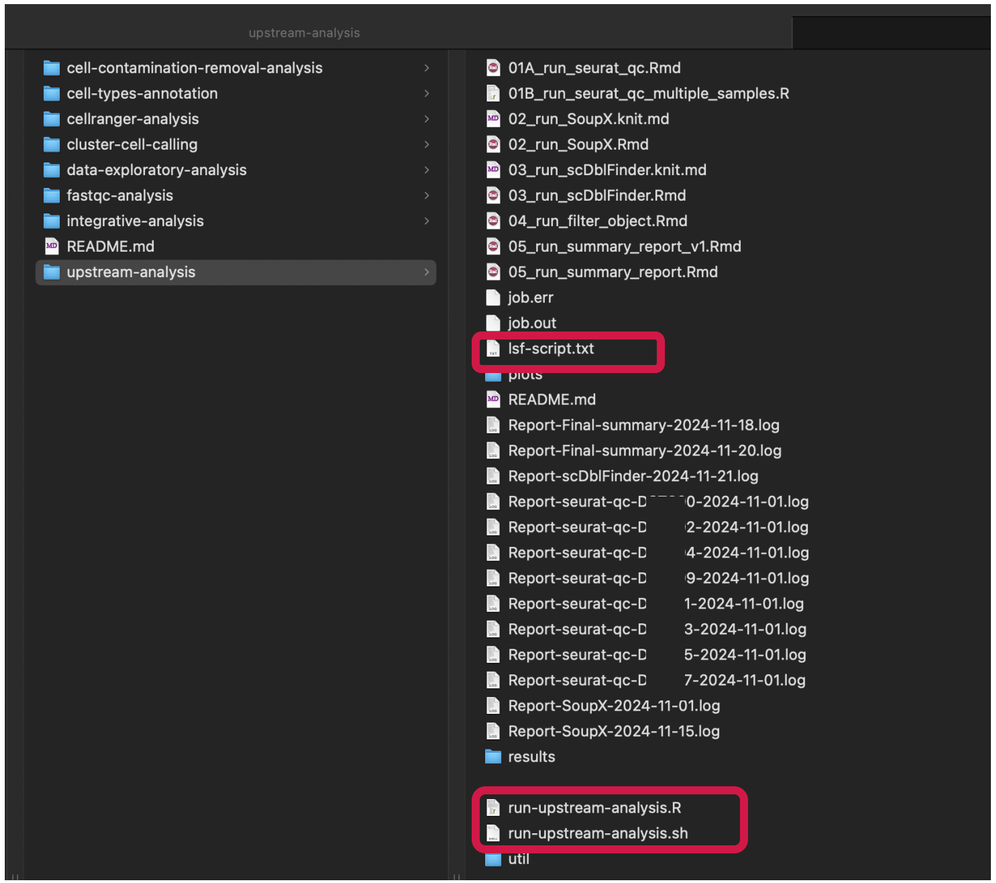


`cellranger-analysis` module



`upstream-analysis` module

description="Pipeline for estimating QC metrics and filtering low quality cells.", required=True



`upstream-analysis` module

README.md

Preview on Save | Preview | Run

Source | Visual

```
# Pipeline for estimating QC metrics for sc-/sn-RNA-Seq Analysis in 10X Genomics data

## Usage

`run-upstream-analysis.sh` is designed to be run as if it was called from this module directory even when called from outside of this directory.

Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `project_parameters.Config.yaml` located at the `root_dir`.


### Run module on an interactive session on HPC within the container

To run all of the Rscripts in this module sequentially on an interactive session on HPC, please run the following command from an interactive compute node (while within the container):

```
bash run-upstream-analysis.sh
```

Run module by using lsf on HPC with the container

There is also the option to run a lsf job on the HPC cluster by using the following command on an HPC node:

```
bsub < lsf-script.txt
```


Folder content

This folder contains scripts tasked to:
(1) Infer QC metrics and associated plots to visually explore the quality of each library of the project.
(2) Evaluate QC metrics and set filters to remove low quality cells in 10x single-cell- and single-nuclei-RNA-sequencing libraries (without cell hashing experiment).

QC Steps and methods

The pipeline allows for the user to include/exclude methods and adjust the pipeline during QC according to the sequence type, expected cells number, type of experiment, and genome reference.

If the user does not wish to include results from step (2), this can be defined in the `project_parameters.Config.yaml` file.

(1) Seurat QC metrics

[Seurat](https://satijalab.org/seurat/articles/pbmc3k_tutorial.html) and [scooter](https://github.com/igordot/scooter) workflows are implemented to pre-process, filter and plot the RNA-sequencing data. The CellRanger output from the `cellranger-analysis` module is used for this step. User will have to define `params` as needed for their experiment.
  - Before and after filter: Plot distribution of the number of genes, UMI, and percent mitochondrial reads per cell.
  - Summary of Cell Statistics: Percent of reads in cells, Median UMI count per cell, Median genes detected per cell, Median percent reads mitochondrial.
  - Data were normalized by using the global-scaling normalization method "LogNormalize" that normalizes the feature expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result. Then, highly variable genes (HVGs) are selected to subset features that indicate high cell-to-cell variation in the dataset (i.e, they are highly expressed in some cells, and lowly expressed in others). Then, these HVGs are used as input to principal component analysis, and the top 30 principal components are selected. A combination of different dimensions (20, 25) and number of neighbors (30, 20, 10) are used along with the principal components to calculate the UMAP (Uniform Manifold Approximation and Projection) embeddings.

Here, the user can select to implement the following strategies to remove low quality cells:
- `step A` [miQC](https://bioconductor.org/packages/devel/bioc/vignettes/miQC/inst/doc/miQC.html) R package. The miQC model is based on the assumption that there are a non-trivial number of compromised cells in the dataset, which is not true in all datasets. If it is already known that the dataset is high-quality with a trivial number of compromised cells, we recommend that the user skip this step.
- `step B` `run_QC_default` function. This is split in two filtering steps.
  - `step 1`: Filter cells with low content of genes expressed and remove mtDNA from each library (as defined in the `params`).
  - `step 2`: `Find_Outlier_Thershold` function. This is an optional step (as defined in the `params`).
If miQC does not identify any low quality cells, then the `step B` will automatically be used as for the filtering strategy.

Post alignment/cell quality filtering parameters
We recommend that the user use the following parameters for initial QC, and then adjust accordingly if necessary:
- `scRNA`: min_genes = 300 (nFeature_RNA (genes detected))
           min_count = 500 (nCount_RNA (UMIs detected))
           mtDNA_pct_default = 15 (ideally 10; Percent Mitochondrial)
- `snRNA`: min_genes = 300 (nFeature_RNA (genes detected))
           min_count = 500 (nCount_RNA (UMIs detected))
           mtDNA_pct_default = 5 (ideally 1; Percent Mitochondrial)

```

```
(2) Estimating and filtering out ambient mRNA (`empty droplets`)

[SoupX](https://cran.r-project.org/web/packages/SoupX/vignettes/pbmcTutorial.html) profiles "the soup", i.e., collection of cell-free mRNAs floating in the input solution. The soup looks different for each input solution and strongly resembles the expression pattern obtained by summing all the individual cells.

SoupX calculates `Cell-specific contamination fraction` (Estimate (or manually set) the contamination fraction, the fraction of UMIs originating from the background, in each cell) and infers a `corrected expression matrix` (Correct the expression of each cell using the ambient mRNA expression profile and estimated contamination).

The CellRanger output from the `cellranger-analysis` module is used for this step.
 - Contamination summary table and Cell-specific contamination fraction plot are generated.

(3) Estimating and filtering out doublets

Popular approach of scRNAseq uses oil droplets or wells to isolate single cells along with barcoded beads. Depending on the cell density loaded, a proportion of reaction volumes (i.e. droplets or wells) will capture more than one cell, forming 'doublets' (or 'multiplets'), i.e. two or more cells captured by a single reaction volume and thus sequenced as a single-cell artifact.

The proportion of doublets is proportional to the number of cells captured. It is common in single-cell experiments to have 10-20% doublets, making accurate doublet detection critical.

Doublets are prevalent in single-cell sequencing data and can lead to artifactual findings. We will use a computational approach to calculate and remove doublets from the library. Here, we use [ScDblFinder](https://bioconductor.org/packages/devel/bioc/vignettes/scDblFinder/inst/doc/scDblFinder.html) method for identifying doublets/multiplets in single-cell data.

The `seurat_obj_raw.rds` object from step (1) is used for this step.
 - Summary table with doublet metrics and doublets prediction plot are generated.

(4) Merging filtered data

Next, we merge count matrices from steps (1-3) after filtering out low quality cells, ambient RNA (optional as defined in the `params`), and doublets. Seurat object and metadata for the library along with UMAP embeddings are saved to be used for downstream analyses.

(5) Final QC summary report

Lastly, we provide a final QC summary report containing graphs and summary tables across each QC step.

Folder structure

The structure of this folder is as follows:

```
├── 01A_run_seurat_qc.Rmd
├── 01B_run_seurat_qc_multiple_samples.R
├── 02_run_SoupX.Rmd
├── 03_run_scDblFinder.Rmd
├── 04_run_filter_object.Rmd
├── 05_run_summary_report.Rmd
├── plots
├── lsf-script.txt
├── README.md
├── results
├── run-upstream-analysis.R
├── run-upstream-analysis.sh
└── util
|   ├── function-calculate-qc-metrics.R
|   ├── function-create-UMAP.R
|   ├── function-process-Seurat.R
|___└── function-run-QC.R
```
```

`upstream-analysis` module

#	steps	aim
1	seurat-qc pipeline	To infer QC metrics and remove low quality cells (miQC R package and/or `run_QC_default`)

```
40
41  ### (1) Seurat QC metrics
42
43  [Seurat](https://satijalab.org/seurat/articles/pbmc3k_tutorial.html) and [scooter](https://github.com/igordot/scooter) workflows are implemented to pre-process, filter and plot the RNA-sequencing data. The
    CellRanger output from the `cellranger-analysis` module is used for this step. User will have to define `params` as needed for their experiment.
44    - Before and after filter: Plot distribution of the number of genes, UMI, and percent mitochondrial reads per cell.
45    - Summary of Cell Statistics: Percent of reads in cells, Median UMI count per cell, Median genes detected per cell, Median percent reads mitochondrial.
46    - Data were normalized by using the global-scaling normalization method “LogNormalize” that normalizes the feature expression measurements for each cell by the total expression, multiplies this by a scale factor
    (10,000 by default), and log-transforms the result. Then, highly variable genes (HVGs) are selected to subset features that indicate high cell-to-cell variation in the dataset (i.e, they are highly expressed in
    some cells, and lowly expressed in others). Then, these HVGs are used as input to principal component analysis, and the top 30 principal components are selected. A combination of different dimensions (20, 25) and
    number of neighbors (30, 20, 10) are used along with the principal components to calculate the UMAP (Uniform Manifold Approximation and Projection) embeddings.
47
48  Here, the user can select to implement the following strategies to remove low quality cells:
49  - `step A` [miQC](https://bioconductor.org/packages/devel/bioc/vignettes/miQC/inst/doc/miQC.html) R package. The miQC model is based on the assumption that there are a non-trivial number of compromised cells in
    the dataset, which is not true in all datasets. If it is already known that the dataset is high-quality with a trivial number of compromised cells, we recommend that the user skip this step.
50  - `step B` `run_QC_default` function. This is split in two filtering steps.
51     - `step 1`: Filter cells with low content of genes expressed and remove mtDNA from each library (as defined in the `params`).
52     - `step 2`: `Find_Outlier_Thershold` function. This is an optional step (as defined in the `params`).
53  If miQC does not identify any low quality cells, then the `step B` will automatically be used as for the filtering strategy.
54
55  #### Post alignment/cell quality filtering parameters
56  We recommend that the user use the following parameters for initial QC, and then adjust accordingly if necessary:
57  - `scRNA`: min_genes = 300 (nFeature_RNA (genes detected))
58             min_count = 500 (nCount_RNA (UMIs detected))
59             mtDNA_pct_default = 15 (ideally 10; Percent Mitochondrial)
60  - `snRNA`: min_genes = 300 (nFeature_RNA (genes detected))
61             min_count = 500 (nCount_RNA (UMIs detected))
62             mtDNA_pct_default = 5 (ideally 1; Percent Mitochondrial)
63
```

`upstream-analysis` module

#	steps	aim
1	seurat-qc pipeline	To infer QC metrics and remove low quality cells (miQC R package and/or `run_QC_default`)
2	SoupX method	To estimate contamination fraction (ambient RNA) and adjust expression matrix

```
63
64  ### (2) Estimating and filtering out ambient mRNA (`empty droplets`)
65
66  [SoupX](https://cran.r-project.org/web/packages/SoupX/vignettes/pbmcTutorial.html) profiles “the soup”, i.e., collection of cell-free mRNAs floating in the input solution. The soup looks different for each input
    solution and strongly resembles the expression pattern obtained by summing all the individual cells.
67
68  SoupX calculates `Cell-specific contamination fraction` (Estimate (or manually set) the contamination fraction, the fraction of UMIs originating from the background, in each cell) and infers a `corrected
    expression matrix` (Correct the expression of each cell using the ambient mRNA expression profile and estimated contamination).
69
70  The CellRanger output from the `cellranger-analysis` module is used for this step.
71   -  Contamination summary table and Cell-specific contamination fraction plot are generated.
72
73
```

`upstream-analysis` module

#	steps	aim
1	seurat-qc pipeline	To infer QC metrics and remove low quality cells (miQC R package and/or `run_QC_default`)
2	SoupX method	To estimate contamination fraction (ambient RNA) and adjust expression matrix
3	scDblFinder method	To identify doublets/multiplets

```
73
74  ### (3) Estimating and filtering out doublets
75
76  Popular approach of scRNAseq uses oil droplets or wells to isolate single cells along with barcoded beads. Depending on the cell density loaded, a proportion of reaction volumes (i.e. droplets or wells) will
    capture more than one cell, forming ‘doublets’ (or ‘multiplets’), i.e. two or more cells captured by a single reaction volume and thus sequenced as a single-cell artifact.
77
78  The proportion of doublets is proportional to the number of cells captured. It is common in single-cell experiments to have 10-20% doublets, making accurate doublet detection critical.
79
80  Doublets are prevalent in single-cell sequencing data and can lead to artifactual findings. We will use a computational approach to calculate and remove doublets from the library. Here, we use
    [ScDblFinder](https://bioconductor.org/packages/devel/bioc/vignettes/scDblFinder/inst/doc/scDblFinder.html) method for identifying doublets/multiplets in single-cell data.
81
82  The `seurat_obj_raw.rds` object from step (1) is used for this step.
83   -  Summary table with doublet metrics and doublets prediction plot are generated.
84
85
```

`upstream-analysis` module

#	steps	aim
1	seurat-qc pipeline	To infer QC metrics and remove low quality cells (miQC R package and/or `run_QC_default`)
2	SoupX method	To estimate contamination fraction (ambient RNA) and adjust expression matrix
3	scDblFinder method	To identify doublets/multiplets
4	Merge and filter data	(steps 2 and 3 are optional during final filttering)
5	Final QC summary report	To create clean object for downstream analysis

```
85
86  ### (4) Merging filtered data
87
88  Next, we merge count matrices from steps (1-3) after filtering out low quality cells, ambient RNA (optional as defined in the `params`), and doublets. Seurat object and metadata for the library along with UMAP
    embeddings are saved to be used for downstream analyses.
89
90  ### (5) Final QC summary report
91
92  Lastly, we provide a final QC summary report containing graphs and summary tables across each QC step.
```

ScRNASeqSnap analysis modules

README.md

Preview on Save | Preview | Run

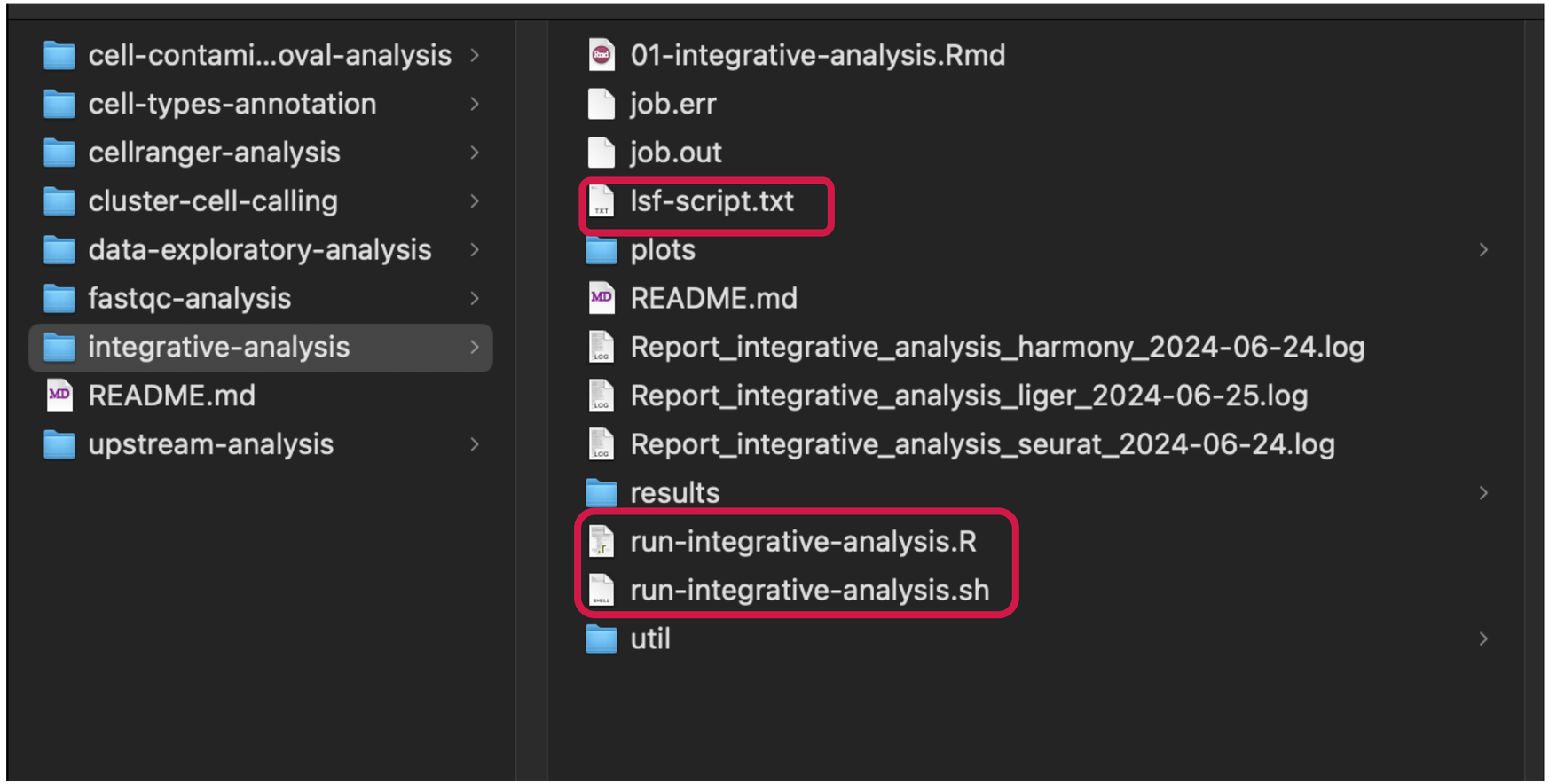
Source | Visual

```
1  # How to use analysis modules in the Single cell RNA Seq Snap workflow (ScRNASeqSnap)
2
3  This repository contains a collection of analysis modules designed to process and analyze single cell and single nuclei RNA (sc/snRNA)
   data from 10X sequencing technology.
4
5  Each module is self-contained and can be executed independently or as part of a larger analysis pipeline. Below is a summary of each
   analysis module, including whether they are required or optional. Furthermore, the analysis modules should be run in the following
   recommended order:
6
7  1. `fastqc-analysis` module (description="Pipeline for FastQC quality control tool for high throughput sequence data analysis.",
   required=True)
8  2. `cellranger-analysis` module (description="Pipeline for running and summarizing Cell Ranger count for single or multiple
   libraries.", required=True)
9  3. `upstream-analysis` module (description="Pipeline for estimating QC metrics and filtering low quality cells.", required=True)
10 4. `integrative-analysis` module (description="Pipeline for Integrative analysis.", required=True)
11 5. `cluster-cell-calling` module (description="Pipeline for cluster cell calling and gene marker analysis.", required=True)
12 6. `cell-contamination-removal-analysis` module (description="To remove clusters and repeat steps (4) and (5), e.g. for PDX
   experiments.", required=False)
13 7. `cell-types-annotation` module (description="Pipeline for annotating cell types.", required=True)
14
15
16 ## Contact
17
18 Contributions, issues, and feature requests are welcome! Please feel free to check
   [issues](https://github.com/stjude-dnb-binfcore/sc-rna-seq-snap/issues).
19
20 ---
21
22 *These tools and pipelines have been developed by the Bioinformatic core team at the [St. Jude Children's Research
   Hospital](https://www.stjude.org/). These are open access materials distributed under the terms of the [BSD 2-Clause
   License](https://opensource.org/license/bsd-2-clause), which permits unrestricted use, distribution, and reproduction in any medium,
   provided the original author and source are credited.*
```

Downstream

`integrative-analysis` module

description="Pipeline for Integrative analysis.", required=True



`integrative-analysis` module

README.md

Preview on Save | Preview

Source | Visual

```
# Pipeline for Integrative analysis for sc-/sn-RNA-Seq Analysis in 10X Genomics data

## Usage

To run the script in this module from the command line sequentially, use:

```
bash run-integrative-analysis.sh
```

`run-integrative-analysis.sh` is designed to be run as if it was called from this module directory even when called from outside of this directory.

Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `project_parameters.Config.yaml` located at the `root_dir`
- `run-integrative-analysis.R`: define `root_dir`, and define params for: `01-integrative-analysis.Rmd` in the `rmarkdown::render code chunk`

Folder content
This folder contains a script tasked to integrate the samples across the project.

The script named `custom_seurat_functions.R` is from [CellGen Programme Template Notebooks](https://github.com/cellgeni/notebooks).

Folder structure

The structure of this folder is as follows:

```
├── 01-integrative-analysis.Rmd
├── lsf_script.txt
├── plots
|   ├── plots_{integration_method}
|   └── Report_integrative_analysis_{integration_method}_<Sys.Date()>.html
├── README.md
├── results
|   ├── metadata_{integration_method}
|   └── seurat_obj_integrated_{integration_method}.rds
├── run-integrative-analysis.sh
└── util
|   ├── custom_seurat_functions.R
|__└── function-samples-integrate.R
```
```

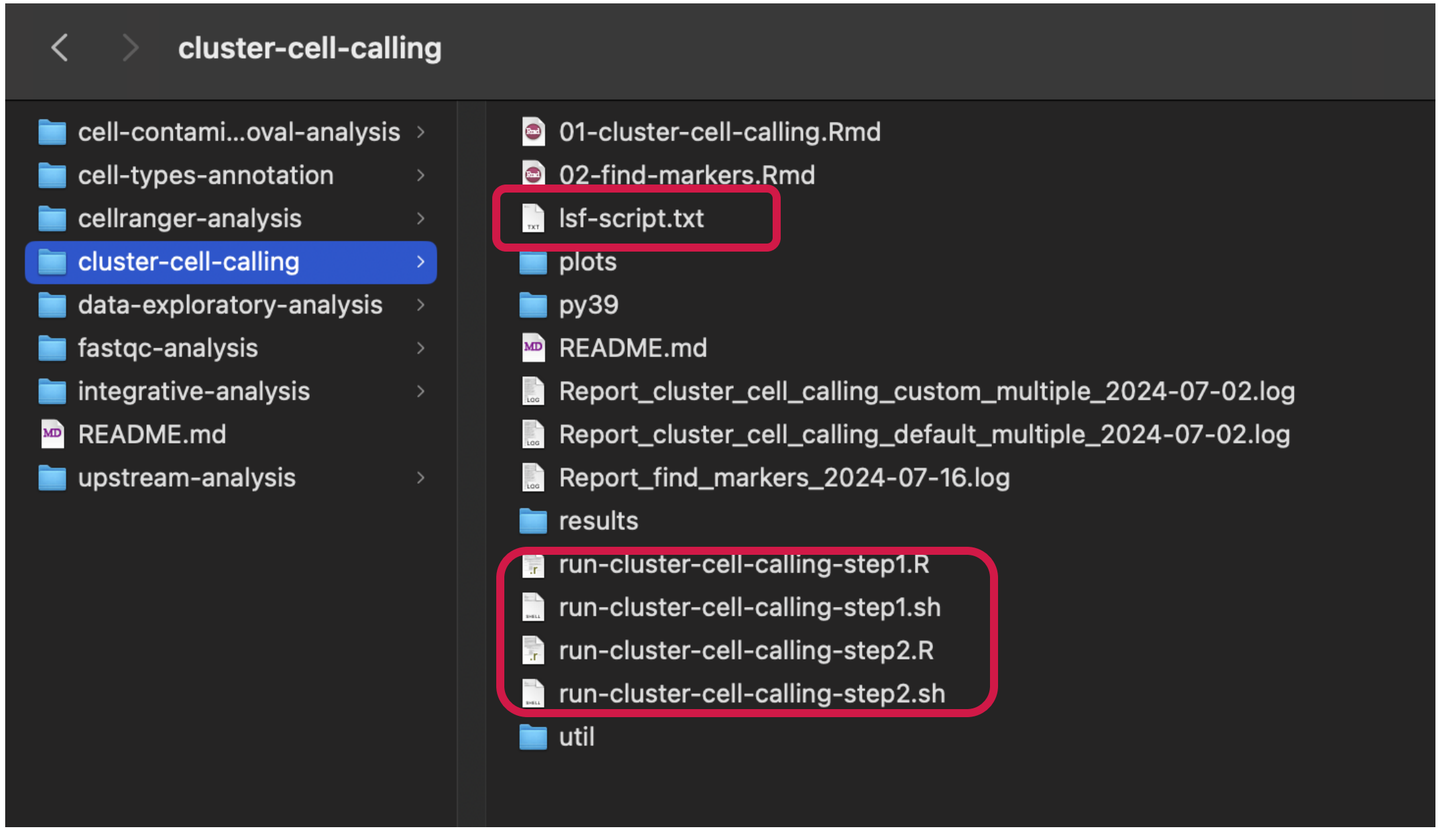
`integrative-analysis` module

#	methods	aim
1	seurat_integration	SCT method for the normalization and variance stabilization of molecular count data from scRNA-seq experiments
2	harmony_integration	Harmony algorithm projects cells into a shared embedding in which cells group by cell type rather than dataset-specific conditions. Harmony simultaneously accounts for multiple experimental and biological factors
3	liger_integration	LIGER algorithm enables the identification of shared cell types across individuals, species, and multiple modalities (gene expression, epigenetic or spatial data), as well as dataset-specific features, offering a unified analysis of heterogeneous single-cell datasets.

Finding cures. Saving children.

`cluster-cell-calling` module

description="Pipeline for cluster cell calling and gene marker analysis.", required=True



Finding cures. Saving children.

`cluster-cell-calling` module

README.md

Preview on Save | Preview | Run

Source | Visual

```
# Pipeline for cluster cell calling and gene marker analysis for sc-/sn-RNA-Seq Analysis in 10X Genomics data

## Usage

To run the script in this module from the command line sequentially, use:

```
bash run-cluster-cell-calling.sh
```

`run-cluster-cell-calling.sh` is designed to be run as if it was called from this module directory even when called from outside of this directory.

Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `project_parameters.Config.yaml` located at the `root_dir`.
- `run-cluster-cell-calling-step1.R`: define `root_dir`, and define params for: `01-cluster-cell-calling.Rmd` in the `rmarkdown::render code chunk`.
- `run-cluster-cell-calling-step2.R`: define `root_dir`, and define params for: `02-find-markers.Rmd` in the `rmarkdown::render code chunk`.

Folder content
This folder contains a script tasked to calculate clusters and find markers for each cluster across the project.

Analysis strategy:
We recommend the user to follow the following steps for running the current module:
- Step (1) `run-cluster-cell-calling-step1.R`: At first, the `01-cluster-cell-calling.Rmd` should be run for a set of resolutions or by default.
- Step (2) `run-cluster-cell-calling-step2.R`: After inspection of the first round of results, the single resolution that fits best the data can be provided and used to run the `02-find-markers.Rmd`.

In order to read the clustertree and decide properly on the resolution, I would consider tissue type, total number of cells in the dataset and expected known cell types in there (i.e., tissue ecosystem). That would help to determine the correct number of clusters in a biologically meaningful way, considering known cell types and avoiding splitting clusters into unstable, small ones. That can also help to explore smaller clusters that might contain unknown or disease/patient-specific clusters that are still worth considering and investigating from the research/clinical perspective.

After inspection of the two rounds of results in the `cluster-cell-calling` module, user can remove clusters if necessary. This is relevant to projects in which we need to remove clusters from the object. This is the case, e.g., in PDX projects, there might be both human and mouse clusters identified after the `02-find-markers.Rmd` step of the the `cluster-cell-calling` module. In this case, we reccomened the user to run the `contamination-remove-cells-analysis` module that allows to remove clusters, repeat normalization and integration steps. This object can then be used for cell type annotation or other type of analysis.

Folder structure

The structure of this folder is as follows:

```
├── 01-cluster-cell-calling.Rmd
├── 02-find-markers.Rmd
├── lsf_script.txt
├── plots
|   ├── 01_cluster_cell_calling_{resolution}
|   ├── 02_find_markers
|   ├── Report_cluster_cell_calling_{resolution}_<Sys.Date()>.html
|   ├── Report_cluster_cell_calling_{resolution}_<Sys.Date()>.pdf
|   ├── Report_find_markers_<Sys.Date()>.html
|   └── Report_find_markers_<Sys.Date()>.pdf
├── README.md
├── results
|   ├── 01_cluster_cell_calling_{resolution}
|   └── 02_find_markers
├── run-cluster-cell-calling-step1.R
├── run-cluster-cell-calling-step2.R
├── run-cluster-cell-calling-step1.sh
├── run-cluster-cell-calling-step2.sh
└── util
|   ├ function-cluster-cell-calling.R
```


Finding cures. Saving children.
```

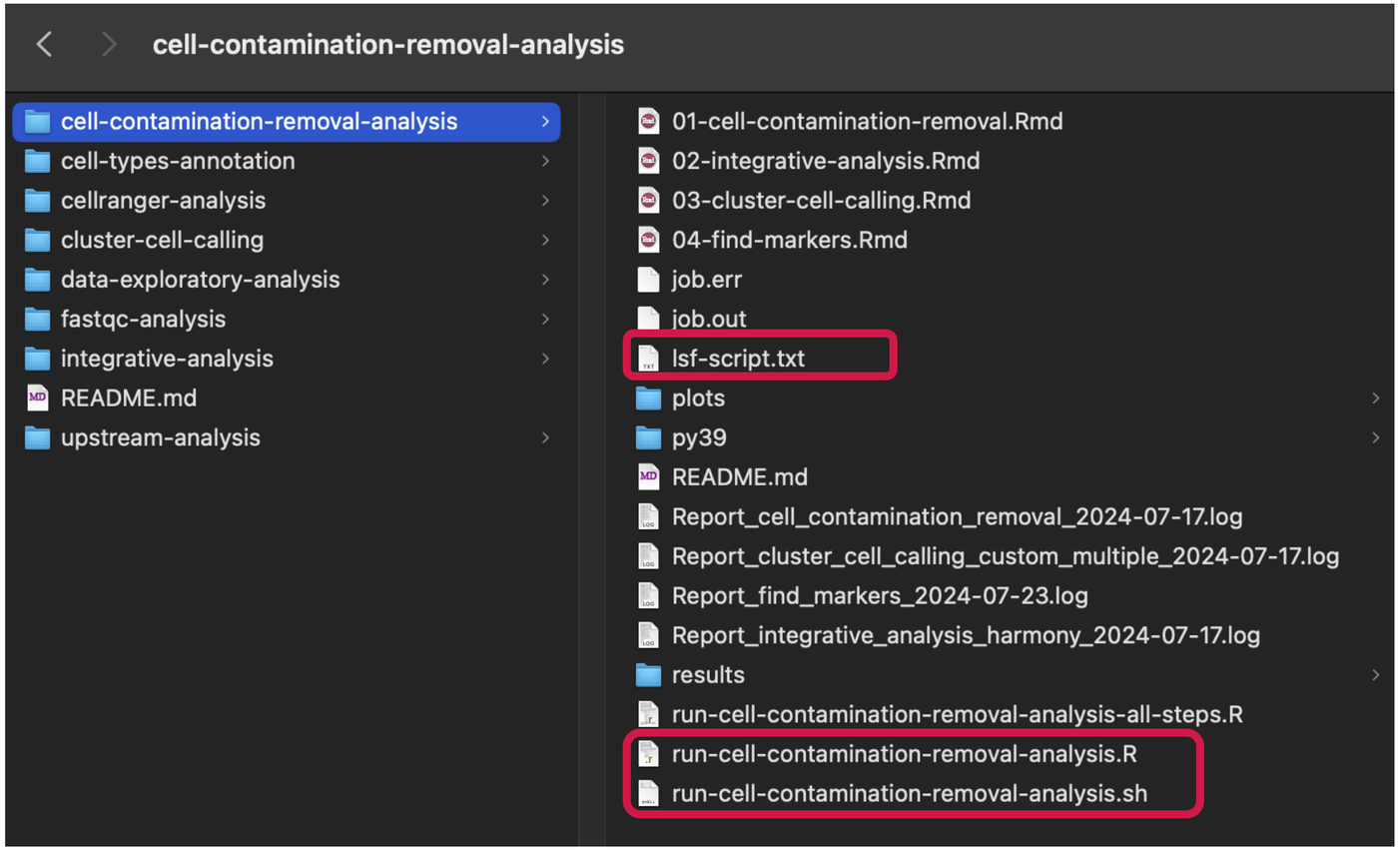
`cluster-cell-calling` module

#	steps	aim
1	cluster-cell-calling	To cluster the cells under different resolutions to define granularity
2	find-markers	To find differentially expressed features (cluster biomarkers)

Finding cures. Saving children.

`cell-contamination-removal-analysis` module - optional

description="To remove clusters and repeat steps (4) and (5), e.g. for PDX experiments.", required=False



`cell-contamination-removal-analysis` module - optional

README.md

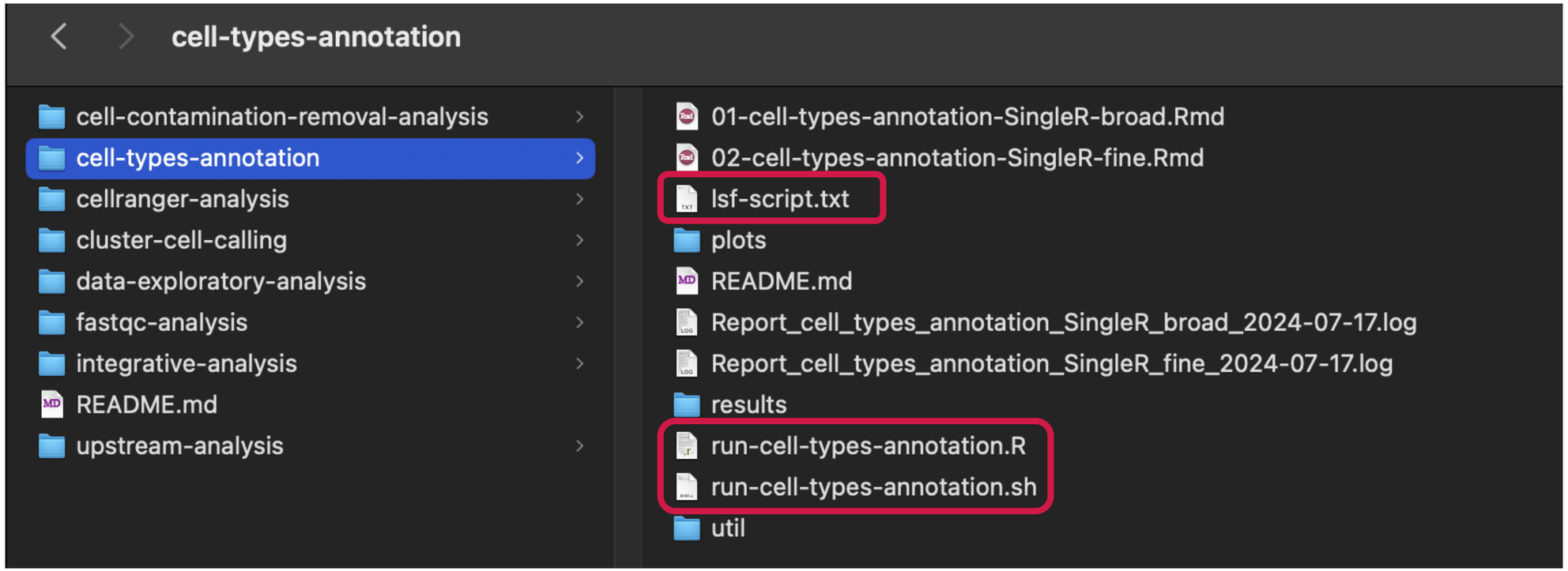
Preview on Save | Preview | Run

Source | Visual

```
# Pipeline for removing cell contamination, and repeating normalization and integration steps analysis for sc-/sn-RNA-Seq Analysis in 10X Genomics data

## Usage

To run the script in this module from the command line sequentially, use:

```
bash run-cell-contamination-removal-analysis.sh
```

`run-cell-contamination-removal-analysis.sh` is designed to be run as if it was called from this module directory even when called from outside of this directory.

Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `project_parameters.Config.yaml` located at the `root_dir`.
- `run-cell-contamination-removal-analysis.R`: define `root_dir`, and define params for: `01-cell-contamination-removal.Rmd` in the `rmarkdown::render code chunk`.

Folder content
This folder contains a script tasked to remove cells and repeat normalization and integration steps analysis. This object can be used for further analysis, e.g., for the `cell-types-annotation` module.

Analysis strategy:

After inspection of the two rounds of results in the `cluster-cell-calling` module, user can remove clusters if necessary. This is relevant to projects in which we need to remove cell contamination from the object, e.g., clusters. This is the case, e.g., in PDX projects, there might be both human and mouse clusters identified after the `02-find-markers.Rmd` step of the the `cluster-cell-calling` module. In this case, we reccomened the user to run the `cell-contamination-removal-analysis` module that allows to remove clusters, repeat normalization and integration steps. This object can then be used for cell type annotation or other type of analysis.

Folder structure

The structure of this folder is as follows:

```
├── 01-cell-contamination-removal.Rmd
├── 02-integrative-analysis.Rmd
├── 03-cluster-cell-calling.Rmd
├── lsf_script.txt
├── plots
    ├── 01_cell_contamination_removal
    ├── 02_integration_{integration_method}
    ├── 03_cluster_cell_calling_{resolution}
    ├── Report_cell_contamination_removal_<Sys.Date()>.html
    ├── Report_integration_{integration_method}_<Sys.Date()>.html
    ├── Report_cluster_cell_calling_{resolution}_<Sys.Date()>.html
    ├── Report_cell_contamination_removal_<Sys.Date()>.pdf
    ├── Report_integration_{integration_method}_<Sys.Date()>.pdf
    └── Report_cluster_cell_calling_{resolution}_<Sys.Date()>.pdf
├── README.md
├── results
    ├── 01_cell_contamination_removal
    ├── 02_integration_{integration_method}
    └── 03_cluster_cell_calling_{resolution}
├── run-cell_contamination_removal-analysis.R
└── run-cell_contamination_removal-analysis.sh
```
```

`cell-types-annotation` module

description="Pipeline for annotating cell types.", required=True



`cell-types-annotation` module

README.md

Preview on Save | Preview | Run

Source | Visual

```
# Pipeline for annotating cell types for sc-/sn-RNA-Seq Analysis in 10X Genomics data

## Usage

To run the script in this module from the command line sequentially, use:

```
bash run-cell-types-annotation.sh
```

`run-cell-types-annotation.sh` is designed to be run as if it was called from this module directory even when called from outside of this directory.

Parameters according to the project and analysis strategy will need to be specified in the following scripts:
- `project_parameters.Config.yaml` located at the `root_dir`
- `run-cell-types-annotation.R`: define `root_dir`, `data_dir`, and define params for: `01-cell-types-annotation-SingleR-broad.Rmd` and `02-cell-types-annotation-SingleR-fine.Rmd` in the `rmarkdown::render code chunk`

Folder content
This folder contains a script tasked to identify cell types annotation with SingleR by using a broad and fine resolution labeling across the project.

• Step 1: cell type annotation based on general cell atlas reference. For now, we will be using ` HumanPrimaryCellAtlasData` and `MouseRNAseqData` from the celldex R package for human and mouse data, respectively.
1. Option with SingleR R package: reference to be by `celldex` R package, or public or in-house
2. Option with Labeltransfer Seurat function: reference to be by `celldex` R package, or public or in-house
3. Option with `Cell type signature` function (Cody)
• Step 2: Evaluation of results `from step 1` -> Identify and remove any cells that are due to contamination -> Save filtered object
• Step 3: Repeat `Step 1`

Folder structure

The structure of this folder is as follows:

```
├── 01-cell-types-annotation-SingleR-broad.Rmd
├── 02-cell-types-annotation-SingleR-fine.Rmd
├── lsf_script.txt
├── plots
|   ├── 01-cell-types-annotation-SingleR
|       ├── 01_annotations_broad
|       └── 02_annotations_fine
|   ├── Report_cell_types_annotation_SingleR_broad_<Sys.Date()>.html
|   ├── Report_cell_types_annotation_SingleR_broad_<Sys.Date()>.pdf
|   ├── Report_cell_types_annotation_SingleR_fine_<Sys.Date()>.html
|   └── Report_cell_types_annotation_SingleR_fine_<Sys.Date()>.pdf
├── README.md
├── results
|   ├── 01-cell-types-annotation-SingleR
|       ├── 01_annotations_broad
|       └── 02_annotations_fine
├── run-cell-types-annotation.R
├── run-cell-types-annotation.sh
└── util
|___└── function-cell-type-fractions.R
```
```

`cell-types-annotation` module

#	steps	aim
1	cell-types-annotation-SingleR-broad	To identify cell types by using SingleR and celldex database and run diagnostics
2	cell-types-annotation-SingleR-fine	To identify cell types by using SingleR and celldex database and run diagnostics

Next steps

- ✅ Pipeline development (and testing phase)
- **Testing phase and code review process**
- Validation phase by using other datasets/experiments
- Launching!

Future analysis modules

- `clone-phylogeny-analysis` module
- `infer-tumor-microenvironment-analysis` module
- `differential-expression-analysis` module
- `trajectories-pseudotime-analysis` module
- `rna-velocity-analysis` module
- `cell-types-annotation` module: using custom atlas reference or gene signature score
- R shiny app

More resources 

- Automation and Reproducibility in Computational Biology
- Documentation and Usability
- GitHub Docs
- Docker for beginners
- Docker for Data Scientists

./resources/links-to-resources.md

Ask me questions!

Tonia
Antonia Chroni, PhD
she/her/hers

Senior Bioinformatics Research Scientist
New York, New York

@AntoniaChroni